VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

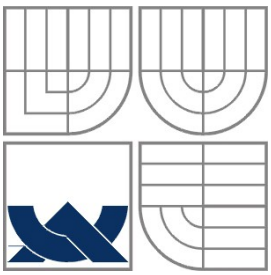
EFEKTIVNÍ TAGOVÁNÍ FOTOGRAFIÍ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

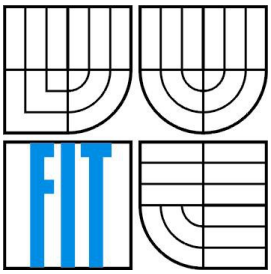
AUTOR PRÁCE
AUTHOR

BC. VÁCLAV PROCHÁZKA

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

EFEKTIVNÍ TAGOVÁNÍ FOTOGRAFIÍ

EFFICIENT IMAGE TAGGING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. VÁCLAV PROCHÁZKA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MICHAL HRADIŠ

BRNO 2013

Abstrakt

Tato práce se zabývá efektivním tagováním fotografií. Konkrétně se zaměřuje na uspořádání jednotlivých fotografií tak, aby tvořily shluky podle svých vlastností a usnadnily tak výběr podobných fotografií, kterým uživatel může efektivně přiřazovat společné tagy zároveň. K tomuto účelu jsou v práci zkoumány známé techniky zobrazování kolekcí fotografií podle jejich vlastností a s tím související metody redukce dimenzionality. Ze zmiňovaných jsou vybrány a otestovány nejvhodnější možnosti. Tato práce navrhuje nový způsob zobrazování kolekcí fortografií na 2D obrazovce, která kombinuje použití časové osy a seskupování podle podobnosti *(Timeline projekce)*. Pro optimální projekci uskupení v mnohorozměrném prostoru příznakových vektorů na 2-rozměrnou obrazovku je v této práci použita metoda redukce dimenzionality nazvaná t-Distributed Stochastic Neighbour Embedding (t-SNE). Jsou popsány různé modifikace t-SNE a způsoby, jak ji kombinovat s časovou osou, a zvolená modifikace je implementována formou webového rozhraní a kvalitativně vyhodnocena experimentem. Na závěr jsou navrženy možnosti pokračování výzkumu.

Abstract

This thesis investigates efficient manual image tagging approaches. It specifically focuses on organising images into clusters depending on their content, and thus on simplifying the selection of similar photos. Such selections may be efficiently tagged with common tags. The thesis investigates known techniques for visualisation of image collections according to the image content, together with dimensionality reduction methods. The most suitable methods are considered and evaluated. The thesis proposes a novel method for presenting image collections on 2D displays which combines a timeline with similarity grouping *(Timeline projection)*. This method utilizes t-Distributed Stochastic Neighbour Embedding (t-SNE) for otpimally projecting groupings in high dimensional feature spaces onto the low-dimensional screen. Various modifications of t-SNE and ways to combine it with the timeline are discussed and chosen combination is implemented as a web interface and is qualitatively evaluated in a user study. Possible directions of further research on the subject are suggested.

Klíčová slova

Tagování fotografií, efektivní tagování, zobrazování kolekcí fotografií, redukce dimenzionality.

Keywords

Image tagging, efficient tagging, viewing of photograph collection, dimensionality reduction.

Citace

Václav Procházka: Efektivní tagování fotografií, diplomová práce, Brno, FIT VUT v Brně, 2013

Efektivní tagování fotografií

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Michala Hradiše. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Václav Procházka
22.5.2013

Poděkování

Děkuji svému vedoucímu práce Ing. Michalovi Hradišovi za jeho podporu a konzultace, které byly jak odborně hodnotné, tak v přátelském duchu.

Obsah

1 Úvod

Cílem této práce je návrh uživatelského rozhraní, které by sloužilo k efektivní práci s menšími i rozsáhlejšími kolekcemi fotografií, tedy pro zefektivnění procházení kolekcí, vyhledávání fotografií a tagování fotografií.

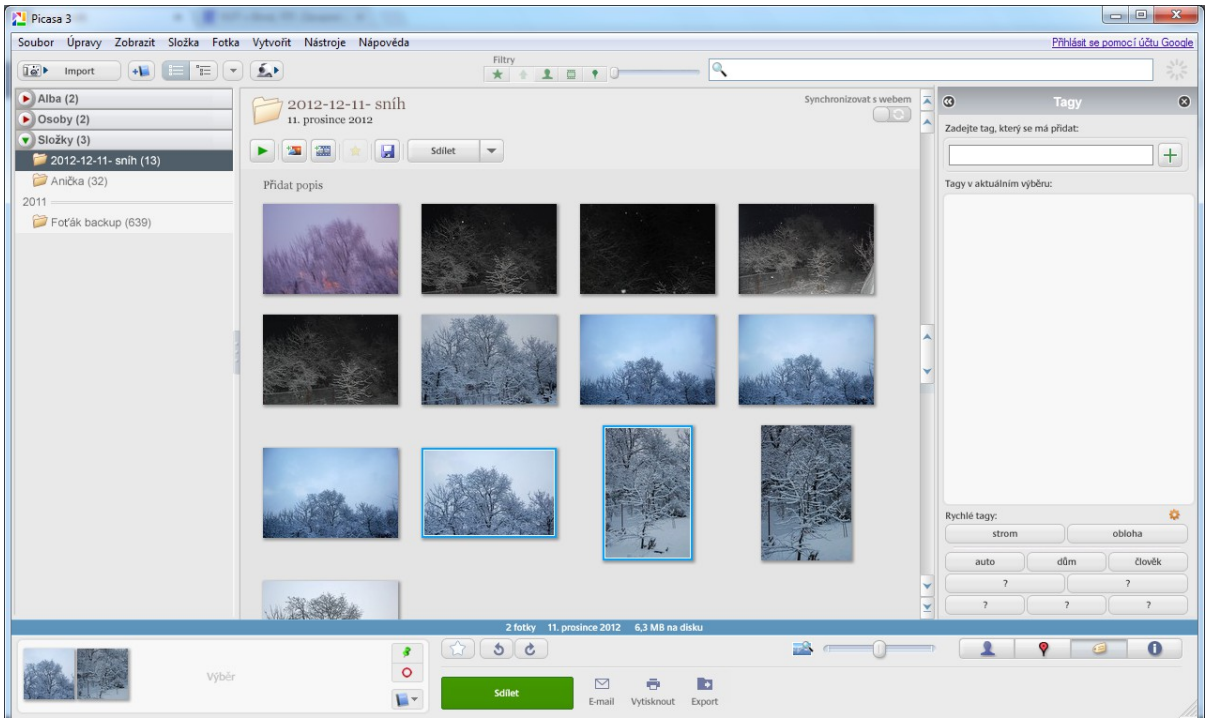
Tyto aspekty spolu navzájem mohou souviset, např. čím je efektivnější procházení kolekcí fotografií, tím bude pravděpodobně efektivnější i jejich tagování, a naopak tagy přiřazené k fotografiím mohou posloužit k organizaci fotografií a k snadnějšímu a přehlednějšímu procházení kolekcí či vyhledávání konkrétních fotografií.

Dnes je nejrozšířenějším schématem uživatelského rozhraní pro tagování fotografií rozdělení obrazovky na větší část, ve které jsou do mřížky uspořádané fotografie, a na část menší, ve které je vstupní řádek pro zadání tagu, který chceme fotografii přiřadit, a ve většině případů i tlačítka s nejčastěji pužívanými tagy (na obrázcích *1.1* a *1.2* jsou zobrazena rozhraní nástrojů *Picasa 3* a *Flickr*). Tato tlačítka mohou v případě některých systémů sloužit i jako prostředek poloautomatického tagování, kdy tlačítka představují tagy, které systém pro danou fotografii navrhl. Rovněž je rozšířená možnost přiřazovat tagy více fotografiím zároveň, s tím, že po výběru většího množství fotek jsou přiřazené tagy odlišeny podle toho, kolik z vybraných fotografií má tento tag přiřazeno (např. v rozhraní *Picasa 3*).

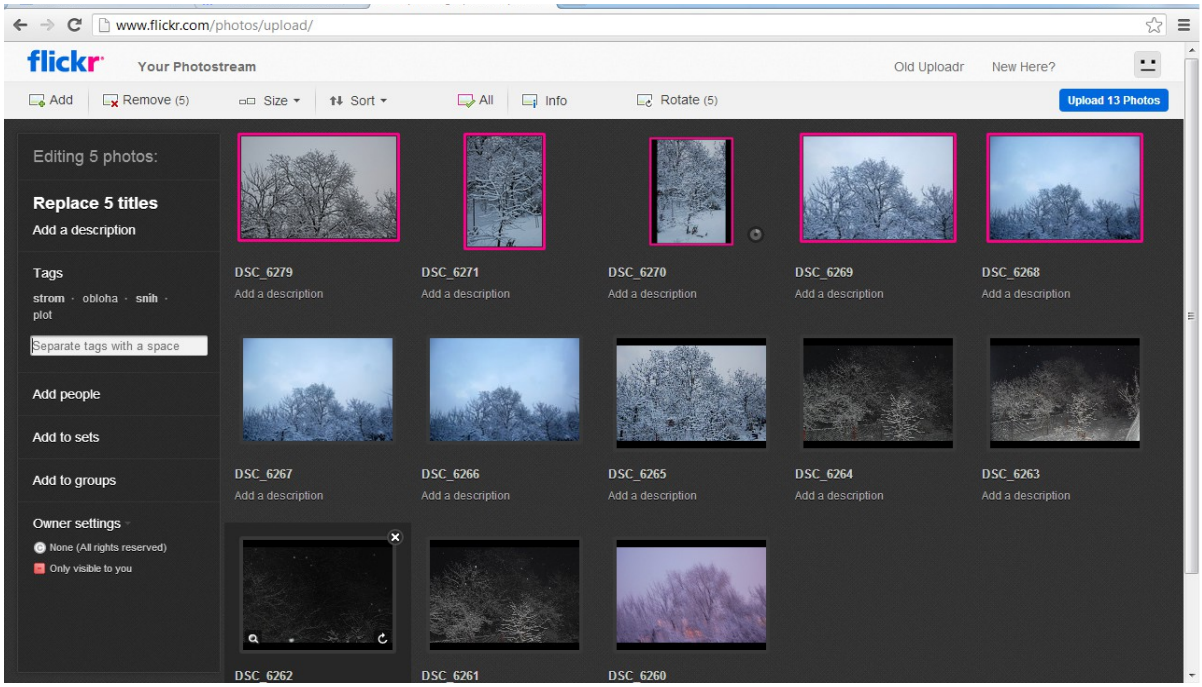
První ze dvou zmiňovaných částí rozhraní, tedy ta pro zadávání tagů, se jeví jako dobře propracovaná, jednoduchá na používání i na pochopení uživatelem, který se se systémem seznamuje. Tlačítek s nabídkou tagů je zpravidla z důvodu přehlednosti použito menší množství. V kombinaci se vstupním řádkem pro textové zadávání tagů je tato část rozhraní jednoduchá a efektivní. Proto se zaměřím spíše na druhou část rozhraní, ve které jsou zobrazeny fotografie kolekce.

Jak bylo řečeno, větší část uživatelských rozhraní dnešních tagovacích systémů tvoří plocha, ve které jsou zobrazeny fotografie kolekce, se kterou uživatel pracuje. Tyto fotografie jsou v dnešních tagovacích systémech zobrazovány výhradně v mřížce, ve které jsou sekvenčně řazeny podle 1 atributu (např. jeden z atributů: název souboru fotografie, datum a čas pořízení fotografie, a jiné), podobně jako při procházení adresáře v počítači. Tento způsob zobrazování fotografií je sice na jednu stranu přehledný, nicméně pokud chce uživatel tagovat efektivně, je potřeba vybrat více fotografií se stejnými vizuálními či sémantickými aspekty a přiřazovat jim společné tagy najednou. Zefektivnění výběru podobných fotografií je možné vhodným rozmístěním fotografií na obrzovce, na které se v této práci zaměřím.

Při výběru způsobu zobrazení fotografií kolekce je v další kapitole nejprve zohledněna práce [7] zabývající se převážně způsoby strukturalizace kolekcí fotografií, dále práce [13] a další, zabývající se metodami redukce dimenzionality, z nichž je pak detalněji popsána metoda, kterou jsem pro své rozhraní zvolil, spolu s důvody této volby. V kapitole *3* pak uvádím svůj návrh uživatelského rozhraní, varianty, které jsem vyzkoušel, důvody pro výběr zvoleného přístupu a bližší popis své implementace. V kapitole *4* pak popisuji a hodnotím výsledky experimentu, který jsem provedl pro vyhodnocení efektivity implementovaného rozhraní. V závěru shrnuji hodnocení dosažených výsledků a přínos mého projektu pro tuto oblast a výhled na další pokračování výzkumu v této oblasti.



Obrázek 1.1 – Ukázka používaného tagovacího rozhraní v nástroji Picasa 3. Ve středu je umístěna mřížka fotografií kolekce, na pravé straně vstupní řádek a tlačítka pro zadávání tagů.



Obrázek 1.2 – Ukázka tagovacího rozhraní v nástroji Flickr (k datu 26.12.2012). Na levé straně vstupní řádek pro zadávání tagů, zbytek rozhraní zabírá mřížka s fotografiemi kolekce.

2 Známé metody zobrazování kolekcí

Tato kapitola se zaměřuje na používané metody zobrazování kolekcí fotografií a popisuje přednosti a nedostatky jednotlivých přístupů, aby pak mohl být zvolen vhodný přístup a případně jeho modifikace, aby lépe vyhovoval specifikovaným požadavkům pro efektivní tagování fotografií, které byly uvedeny v úvodu práce.

Klasický sekvenční způsob procházení, podobný procházení adresáře v počítači, kde jsou fotografie řazené jen podle jména či data, je z hlediska přehlednosti efektivní jen pro velmi malé kolekce fotografií. V případě větších kolekcí fotografií se nabízejí jiné vhodnější způsoby zobrazování kolekcí, které využívají vlastnosti fotografií.

Jeden z možných vhodných způsobů zobrazení kolekce fotografií je seskupování fotografií na obrazovce počítače podle vhodně zvolených vlastností fotografií (histogramy, nastavení fotoaparátu v době pořízení a jiné), tak, aby fotografie, které tyto vlastnosti mají podobné, byly blíže u sebe než fotografie, které se v těchto vlastnostech liší. V takovém případě se tyto vlastnosti reprezentují pomocí příznakového vektoru (feature-vector, FV). Takovýto vektor je vypočítán pro každou fotografii a porovnáním vektorů různých fotografií lze určit míru podobnosti fotografií. Takovouto obecně N-dimenzionální informaci o vzájemné podobnosti fotografií je následně potřeba vhodně přenést na 2-rozměrnou obrazovku počítače tak, aby byly co nejvíce zachovány vzdálenosti mezi fotografiemi tak, jak byly v původním N-rozměrném prostoru. Toto zajistí, že 2-rozměrná reprezentace bude co nejvěrněji zachycovat podobnost fotografií na základě vlastností, zachycených v příznakových vektorech (viz [7]).

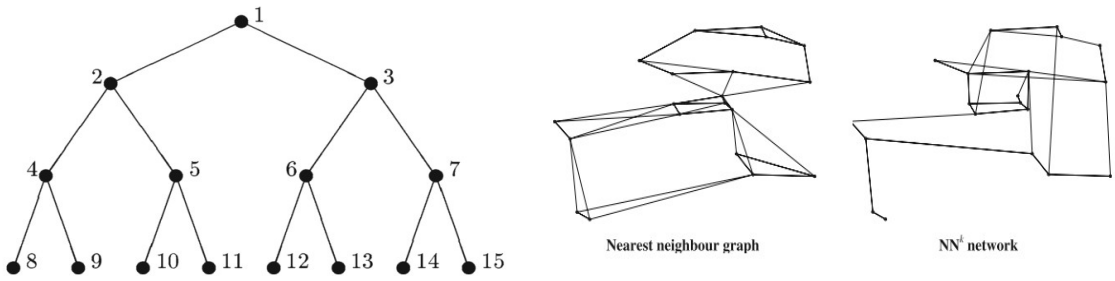
Jak uvádí [7] v práci zabývající se procházením kolekcemi fotografií, existuje několik obecně rozdílných způsobů, jak přistupovat k reprezentaci informací z příznakových vektorů na 2-rozměrné obrazovce počítače. Zmiňovaná práce se zaměřuje na strukturální reprezentaci podobností pomocí hierarchií či sítí samostatně nebo v kombinaci s interaktivní redukcí dimenzionality příznakových vektorů.

2.1 Statické hierarchie

Prvním známým způsobem strukturalizace kolekce fotografií je podle [7] tvorba statické hierarchie, čímž je myšleno, že hierarchie je vytvořena při vložení kolekce do systému, a při procházení touto strukturou v ní samé nedochází k žádným změnám.

Statická hierarchie (strom, viz obr. *2.1.a*) je tvořena pomocí shlukování ([5], [1]), buď pomocí aglomerativního nebo divizního. Divizní shlukování má sice lineární složitost, což je v porovnání s kvadratickou složitostí aglomerativního výhodou, ale na druhou stranu aglomerativní shlukování poskytuje ve výsledku daleko intuitivnější uskupení. Z tohoto důvodu je i přes svou výpočetní složitost používáno spíše aglomerativní shlukování.

Výhodou hierarchické organizace kolekce fotografií je zužování výběru od obecného ke specifičtějšímu, což uživateli poskytuje pocit postupného upřesňování skupiny vybraných fotografií. Na druhou stranu nevýhodou je fakt, že hierarchie umožňuje pouze vertikální pohyb stromovou strukturou, což může v uživateli vést k pocitu omezení prohledávání. Další nevýhodou je obtížná volba fotografií, reprezentujících obsah shluku. Centroidy shluků zřídka poskytují věrné shrnutí informací o všech fotografiích daného shluku.



a) *b)* *c)*

Obrázek 2.1 – Rozdílné uspořádání kolekce fotografií pomocí statických struktur:
a) hierarchie (strom) – neexistuje horizontální propojení mezi sousedy
b) Nearest Neighbour síť - oproti NN^k větší množství propojení
c) NN^k síť. - oproti b) méně propojení. Neposkytuje kvalitní globální nadhled.
Schémata převzata z [7].

2.2 Statické sítě

Statické sítě jsou další možností strukturální organizace fotografií v kolekci (obrázek *2.1.b* a *c*). Vhodnými způsoby tvorby sítě pro účely procházení a tagování kolekce fotografií jsou algoritmy *Nearest Neighbour* (NN, [3], [4]), který propojí hranami (přechody pro procházení) uzly (fotografie), jejichž příznakové vektory jsou si nejblíže. Nebo tzv. *NN^k Sítě* ([6]), ve kterých jsou s danou fotografií propojeny fotografie s nejmenší vzdáleností v každé dimenzi příznakového vektoru, což poskytuje nezaujatost vůči různým vizuálním příznakům.

Navigace v sítích obecně je mnohem méně omezená, než v hierarchiích, ale zároveň je mnohem složitější poskytnout uživateli kvalitní globální nadhled nad kolekcí. Sítě jsou používány v různých odvětvích lidské činnosti velmi zřídka, častěji jsou využívány hierarchie. Jednak proto, že hierarchie nabízí právě onen nadhled a možnost indexace a snadného přístupu, a dále proto, že hierarchie je vždy možné zobrazit v podobě planárního grafu (tedy grafu nakresleného v rovině, kde se žádné hrany neprotínají), zatímco u sítí, kromě těch naprosto triviálních, toto není možné, čímž jsou pro člověka hůře použitelné.

2.3 Dynamické struktury

Dynamické struktury jsou hybridním způsobem mezi klasickým procházením kolekcí a tzv. Query By Example (QBE) přístupem, kdy je kolekce prohledávaných objektů zúžena určením některých parametrů, které by měl hledaný objekt splňovat.

Prvním takovým přístupem je *Ostensivní procházení* ([2]), který uživateli dává pocit, že dynamicky rozkrývá stromovou strukturu. Ve skutečnosti však výběrem z nabídnutých fotografií zpřesňuje podmínky, které má splňovat hledaná fotografie či skupina fotografií. Podmínky po výběru fotografie jsou vážená suma příznakového vektoru právě vybrané fotografie a fotografií vybraných v předchozích krocích.

Dalším způsobem je *Redukce dimenzí*, (použita např. v [12]) kdy jsou např. centroidy shluků některou z existujících metod projekce promítnuty z N-rozměrného prostoru příznakových vektorů do 2-rozměrného prostoru obrazovky. Při výběru jednoho z centroidů daného shluku se zobrazí k danému centroidu nejbližší fotografie.

Poslední přístup, který si zde uvedeme, je nazván *Clustering search results* (Výsledky shlukovacího vyhledávání, [18]). Uživatelé vybírají z „aktivní množiny“ fotografií, která na začátku obsahuje všechny fotografie kolekce. Systém pak seřadí fotografie množiny podle jejich podobnosti k právě vybrané fotografii a ponechá v aktivní množině jen určité fixní procento nejpodobnějších fotografií. Nevýhodou přístupu je omezená svoboda procházení kolekce, protože jakmile se z aktivní množiny některá fotografie odstraní není možné ji dalšími výběry dostat do množiny zpátky.

Dynamické metody tedy nabízí jistou adaptaci na potřeby uživatele v době, kdy se systémem pracuje, na druhou stranu potřebují za běhu čas na provádění dalších výpočtů. Jak uvádí [7], použití dynamických metod na rozsáhlejší kolekce fotografií ještě nebylo dostatečně prozkoumáno, ale je jasné, že by bylo potřeba další optimalizace algoritmů.

2.4 Metody projekce

Při použití libovolné struktury z výše uvedených bude vždy potřeba vhodně zobrazit určité množství aktuálně zobrazovaných fotografií na dvourozměrnou obrazovku počítače tak, aby byly co nejlépe zachovány vzdálenosti mezi jejich reprezentacemi příznakovými vektory v původním vícerozměrném prostoru. Je tedy potřeba použít projekci, která tuto podmínku splní do míry potřebné pro efektivní tagování fotografií.

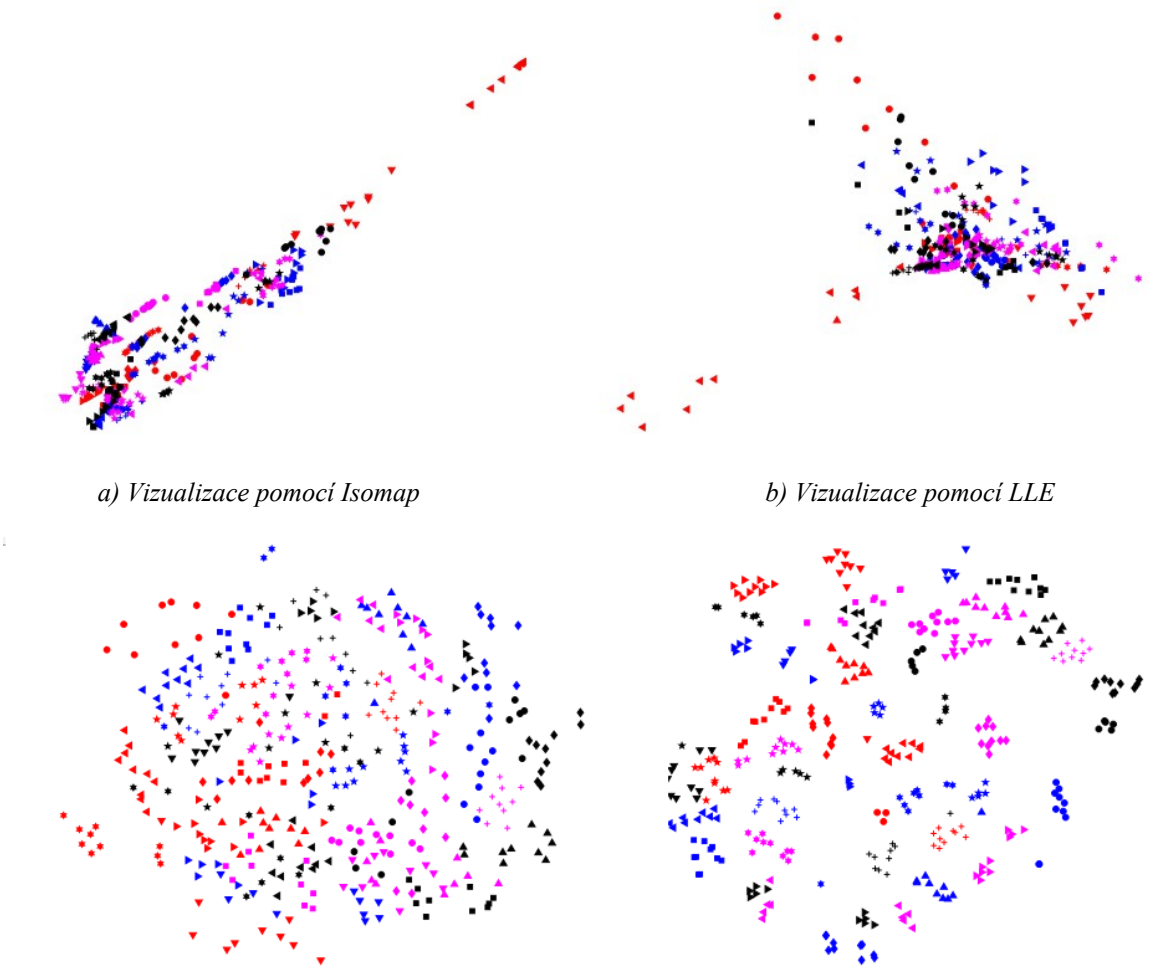
Metod projekce existuje poměrně velké množství, z těch známějších např. Principal Component Analysis (PCA) [9], Multi-Dimensional Scaling (MDS) [17], Sammonovo mapovaní [15], Stochastic Neighbour Embedding (SNE) [8], Isomap [16], Localy Linear Embedding (LLE) [14].

Jak uvádí [13], lineární metody, jakými jsou PCA a MDS nejsou pro účely procházení kolekce fotografií příliš vhodné. Zaměřují se na to, aby body (v našem případě příznakové vektory, reprezentující fotografie), které si v původním prostoru nejsou podobné, byly do dvojrozměrného prostoru promítnuty co nejdál od sebe. Pro zobrazování fotografií je důležitější, aby u fotografií, které jsou si v původním prostoru velmi podobné, byly blízko u sebe jejich průměty do 2-rozměrného prostoru, což typicky není možné u lineárních metod jakými jsou PCA a MDS, jak uvádí [13].

Další ze zmiňovaných metod (Sammonovo mapování, SNE, Isomap, LLE) už lineárními metodami nejsou. Článek [13] uvádí, že jsou sice úspěšné pro projekci s umělými datasety, ale při použití nad reálnými datasety už příliš úspěšné nejsou – většina z těchto technik není schopná udržet zároveň lokální i globální strukturu dat, tedy zachovat vzájemné uspořádání na různém měřítku v rámci jedné projekce.

Ze známých metod se jako nejvhodnější jeví metoda projekce t-SNE navržená v práci [13]. Jedná se o modifikaci metody SNE tak, aby dokázala věrněji zachovat lokální i globální strukturu dat, například přítomnost shluků v různém měřítku.

Rozdíl výsledků mezi ostatními metodami a metodou t-SNE je dobře demonstrován na obrázku *2.2* převzatém z [13], který znázorňuje projekci příznakových vektorů fotografií z datasetu Olivetti (Olivetti faces dataset) obsahujícího fotografie tváří 40 osob, pro každou osobu více snímků pořízených v různou dobu, s různým osvětlením a různými podmínkami. Z tohoto obrázku je patrné, že t-SNE poskytuje ve svých výsledcích mnohem jasněji ohraničené shluky, než je tomu u srovnávaných metod (Sammonovo mapování, Isomap, LLE). Tyto jasně ohraničené shluky jsou pro navrhované uživatelské rozhraní pro tagování fotografií vhodné, protože uživateli usnadní výběr skupiny podobných fotografií, kterým bude potenciálně přiřazovat stejné tagy. Tuto metodu jsem si z tohoto důvodu zvolil pro navrhované uživatelské rozhraní a podrobněji ji popíši v následující kapitole.



a) Vizualizace pomocí Isomap *b) Vizualizace pomocí LLE*

c) Vizualizace pomocí Sammonova mapování *d) Vizualizace pomocí t-SNE*

Obrázek 2.2 – Projekce příznakových vektorů fotografií datasetu Olivetti obsahujícího fotografie tváří (Olivetti faces dataset), zdroj [13]. *Každý tvar a barva znázorňují projekce fotografií tváře jedné osoby, tedy by měly tvořit jeden shluk.*

2.5 Projekce t-SNE

V této kapitole podrobněji popíšu projekční metodu t-SNE (t-Distributed Stochastic Neighbour Embedding, [13]), kterou dále používám pro navrhované tagovací uživatelské rozhraní. Tato metoda vychází z metody SNE [8].

Nejprve, stejně jako metoda SNE, převede mnohodimenzionální euklidovské vzdálenosti mezi body v původním prostoru do podmíněných pravděpodobností, které reprezentují podobnost. Podobnost bodů x_j a x_i je podmíněná pravděpodobnost $p(j|i)$, že x_i by si vybralo x_j za svého souseda, pokud by sousedé byli vybíráni na základě jejich hustoty pravděpodobnosti pod Gaussovou křivkou se středem v x_i. Pro blízké body bude $p(j|i)$ relativně vysoká, zatímco pro vzdálené body se bude blížit nekonečně malým hodnotám, pokud je vhodně zvolena hodnota směrodatné odchylky Gaussovy křivky σ. Matematicky je podmíněná pravděpodobnost $p(j|i)$ dána následujícím vztahem:

$$p(j|i) = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k=0, k\neq i}^{n} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}, \quad \textbf{(1)}$$

kde *n* je počet všech zpracovávaných bodů. Hodnota směrodatné odchylky σ_i Gaussovy křivky se liší pro každý bod v původním mnohorozměrném prostoru, protože hustota bodů v původním prostoru je proměnlivá. V hustších oblastech je vhodnější menší hodnota σ_i, aby došlo k dostatečnému odlišení míry podobnosti fotografií. V řidších naopak hodnota větší, aby i vzdálenější body v řidších oblastech tvořily shluky. Každá konkrétní hodnota σ_i tvoří konkrétní rozložení hustoty pravděpodobnosti P_i přes všechny body a toto rozložení má entropii přímo úměrnou hodnotě σ_i. V této fázi se pro každý mnohorozměrný bod provede binární vyhledávání vhodné hodnoty σ_i, která nad daným bodem vytvoří P_i s takovou entropií, aby její *perplexita* odpovídala fixní *perplexitě* zadané uživatelem. *Perplexita* je vnějším parametrem zadaným uživatelem, který může být interpretován jako plynulá míra počtu sousedů, které jsou pro každý bod uvažovány. Typické hodnoty se nachází mezi 5 a 50.

Hodnota perplexity pro rozložení s aktuální směrodatnou odchylkou σ_i se spočítá pomocí vzorce (2):

$$Perp(P_i) = 2^{H(P_i)}, \quad \textbf{(2)}$$

kde $H(P_i)$ je Shannonova entropie P_i počítaná podle vztahu (3):

$$H(P_i) = \log(\sum_j p(j|i)) + \frac{\sigma_i \cdot \sum_j \|x_i - x_j\| \cdot p(j|i)}{\sum_j p(j|i)} \quad \textbf{(3)}$$

Poté, co je binárním vyhledáváním nalezena hodnota σ pro každý bod, je podle vzorce (1) vypočtena podmíněná pravděpodobnost $p(j|i)$ pro všechna *i* a *j*. Tato podmíněná pravděpodobnost je pak symetrizací převedena na složenou pravděpodobnost p_{ij} podle vzorce

$$p_{ij} = \frac{p(j|i) + p(i|j)}{2n}, \quad \textbf{(4)}$$

kde *n* je počet všech bodů, které chceme promítnout. Tím metoda docílí toho, aby pro všechny body x_i platilo $\sum_j p_{ij} > \frac{1}{2n}$, v důsledku čehož každý bod x_i významně přispěje k níže uvedené cenové funkci, určující míru věrnosti 2-rozměrného průmětu vůči původním mnohorozměrným datům.

Následuje *náhodná* inicializace průmětů bodů do 2-rozměrného prostoru a iterativní část algoritmu. V této části jsou v cyklu vždy nejprve vypočteny mezi všemi promítnutými body ve 2-rozměrném prostoru složené pravděpodobnosti q_{ij}, že jsou dané body v tomto prostoru svými sousedy. Tuto pravděpodobnost metoda t-SNE počítá pomocí Studentova t-rozložení (Student t-distribution), které na rozdíl od Gaussovského rozložení vyruší nechtěné přitažlivé síly mezi průměty bodů, které reprezentují jen středně podobné body původního prostoru. Navíc je oproti Gaussovskému rozložení snažší k výpočtu, neobsahuje totiž exponenciální funkci, což je značná výhoda, protože složená pravděpodobnost q_{ij} je během algoritmu počítána pro všechny body mnohokrát. Autoři používají Studentovo t-rozložení s jedním stupněm volnosti, složená pravděpodobnost q_{ij}, tedy podobnost mezi 2-rozměrnými průměty, se potom vypočítá podle následujícího vztahu:

$$q_{ij} = \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1+\|y_k - y_l\|^2)^{-1}}, \quad \textbf{(5)}$$

kde body y představují souřadnice aktuálních průmětů vícedimenzionálních bodů do 2-rozměrného prostoru.

Dále je v každém cyklu vypočten gradient cenové funkce - Kulbach-Leiblerovy divergence mezi podobnostmi mezi body v původním prostoru a podobnostmi mezi odpovídajícími body promítnutými, tedy divergence mezi pravděpodobnostmi p_{ij} a q_{ij}. Cenová funkce, tedy Kulbach-Leiberova divergence je počítána podle vztahu (6), její gradient podle vztahu (7).

$$C = KL(P\|Q) = \sum_i \sum_j p_{ij} \log p_{ij} - p_{ij} \log q_{ij} \quad \textbf{(6)}$$

$$\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1+\|y_i - y_j\|^2)^{-1} \quad \textbf{(7)}$$

Cílem algoritmu je umístit průměty tak, aby jejich podobnosti co nejlépe odpovídaly podobnostem odpovídajících bodů v původním prostoru, takže se snaží Kulbach-Leiblerovu divergenci těchto podobností minimalizovat. Uvedený gradient je pak použit ke změně pozice průmětu bodu podle následujícího vztahu, aby byla divergence zmenšena a tudíž bylo dosaženo lepší věrnosti uspořádání průmětů vzhledem k uspořádání bodů v původním prostoru:

$$y_i^{(t)} = y_i^{(t-1)} + \alpha \cdot (y_i^{(t-1)} - y_i^{(t-2)}) - \eta_i \cdot \frac{\delta C}{\delta y_i}, \quad \textbf{(8)}$$

kde $y_i^{(t)}$, $y_i^{(t-1)}$, $y_i^{(t-2)}$ jsou po řadě nová pozice průmětu, pozice průmětu v minulém a předminulém kroku (iteraci) metody, α je moment setrvačnosti a η_i je učící rychlost, která je na začátku nastavena na uživatelem zvolenou hodnotu a v průběhu výpočtu je aktualizována prostředky schématu pro adaptivní rychlost učení popsaném v [11]. Konkrétně v případě, že nový gradient změní pozici průmětu podobným směrem jako v kroku minulém, tak je učící rychlost η_i daného průmětu zvětšena o 20% své *inicializační velikosti*, v případě opačném je zmenšena o 20% své *aktuální velikosti*.

Tímto způsobem algoritmus iterativně pokračuje, dokud není změna Kulbach-Leiblerovy divergence mezi jednotlivými kroky menší, než nastavený práh, nebo dokud není dosaženého určitého počtu iterací.

Autoři ve své práci zmiňují dvě optimalizační techniky, *ranou kompresi* a *rané zveličování*. *Raná komprese* znamená, že jsou náhodné inicializační průměty do 2-rozměrného prostoru provedeny na velmi malém prostoru, což umožní snazší pohyb shluků skrz sebe navzájem, což dá metodě prostor k nalezení co nejlepší globální organizace průmětů. *Rané zveličování* znamená uměle zvětšit hodnoty p_{ij} na několikanásobek původní velikosti, např. vynásobit všechny tyto hodnoty konstantou 4, v určitém množství počátečních iterací, což způsobí, že promítnuté body budou mít větší tendenci tvořit sevřené a široce ohraničené shluky, což zanechá v 2-rozměrném prostoru dost volného místa, aby se shluky mohly navzájem mezi sebou pohybovat a najít si co nejvhodnější pozici pro zachování globální struktury bodů v původním prostoru.

Metoda t-SNE ve výsledku dbá jak na modelování velmi nepodobných bodů v původním prostoru velkými vzdálenostmi jejich projekcí do 2D, tak na modelování podobných bodů v původním prostoru velmi malými vzdálenostmi jejich projekcí ve 2D a tím poskytuje mnohem uspokojivější výsledky, než ostatní známé metody (viz obrázek *2.2*). Rovněž je patrné, že je schopná mnohem uspokojivější tvorby shluků, které jsou kompaktní a zřetelně oddělené při vhodně zvolených parametrech metody.

3 Uživatelské rozhraní pro anotaci

V této kapitole popisuji svůj návrh uživatelského rozhraní pro efektivní tagování fotografií. Základní přístup k návrhu jsem zvolil stejný jako v případě rozhraní nástrojů *Picasa 3* a *Flickr*, tedy rozdělení obrazovky na 2 části.

První část zabírá menší díl obrazovky a slouží pro přiřazování tagů vybrané fotografii či skupině fotografií. Jak uvádím v úvodu, tato část se jeví dobře propracovaná a použitelná, takže se v této práci zaměřuji spíše na druhou ze dvou hlavních částí uživatelského rozhraní.

Touto druhou částí, zabírající větší díl obrazovky, je plocha, na které jsou zobrazeny fotografie kolekce, se kterými uživatel v dané chvíli pracuje. V systémech a nástrojích, které jsou dnes rozšířené (např. zmiňované nástroje *Picasa 3* a *Flickr*), jsou tyto fotografie zobrazované do pravidelné mřížky, do které jsou umístěny sekvenčně většinou podle názvu souboru fotografie. Při návrhu této části svého uživatelského rozhraní hledám jiný vhodnější způsob zobrazení kolekce fotografií, než je uvedená mřížka, takový, aby uživateli co nejvíce usnadnil orientaci v kolekci a co nejvíce zefektivnil přiřazování tagů fotografiím. K tomuto účelu používám metodu redukce dimenzionality (projekci) t-SNE podrobněji popsanou v poslední části předchozí kapitoly, která se pro mé účely hodí nejvíce, z důvodů uvedených v předchozí kapitole.

V následujících částech této kapitoly nejprve popisuji použité příznakové vektory fotografií. Tyto vektory jsou použity jako vstup projekce t-SNE. V dalších částech pak popisuji možnosti použití této projekce pro zobrazení kolekce fotografií, které jsem implementoval a otestoval, a uvádím důvody pro volbu přístupu, který jsem nakonec zvolil.

3.1 Příznakové vektory

Vlastnosti fotografií reprezentuji pomocí příznakových vektorů, které se skládají z několika částí. První částí příznakového vektoru, a také nejobjemnější, jsou hodnoty barevného histogramu, spočítaného způsobem, který je popsán v práci [10]. Jedná se o 4098 hodnot v každém příznakovém vektoru. Tato část slouží k reprezentaci vizuálních aspektů fotografie a je základem našich příznakových vektorů. Samostatně ji normalizuji jako 4098-rozměrný vektor.

Další částí jsou informace získané z EXIF informací souboru fotografie. Z těchto informací používám nejprve délku expozice, nastavenou citlivost senzoru fotoaparátu podle stupnice ISO a clonové číslo. Kombinace těchto informací, za předpokladu, že hodnoty byly při focení nastaveny vhodně, vnáší zpětně do příznakového vektoru informaci o kvalitě osvětlení scény, a mohou tak přispět k odlišení fotografií focených ve dne nebo v noci, v místnosti nebo venku. Každá z těchto hodnot (expozice, ISO, clona) je oddělené normalizována v rámci celé kolekce fotografií do intervalu od 0 do 1.

Dále je s EXIF informací souboru fotografie použito datum a čas pořízení fotografie – zcela jistě vhodné k odlišení denní doby pořízení fotografie, stejně tak i ročního období. Tyto informace ukládám do všech příznakových vektorů ve dvou formátech. Nejprve v původním formátu jako řetězec „YY:MM:DD HH:MM:SS“, tyto informace slouží pro seřazení fotografií chronologicky podle pořízení a pro případné popisky zobrazované v uživatelském rozhraní.

Druhý formát uložení času pořízení fotografie slouží k určení denní doby a ročního období a tvořím ho ze vstupní informace v původním formátu řetězce „YY:MM:DD HH:MM:SS“ tak, že nejprve oddělím informace na 2 části: první část vztahující se k části dne, ve které byla fotografie pořízena (hodiny, minuty, sekundy) a druhou část vztahující se k části roku (měsíce a dny). Rok pořízení fotografie zde zanedbávám, tato informace je uložena ve formátu popsaném v předchozím

odstavci. Nyní obě zmiňované části, čas roku t_{rok} i čas dne t_{den}, přepočtu každou do jedné číselné hodnoty normalizované do intervalu od 0 do 1 podle následujících vztahů:

$$t_{rok} = \frac{31 \cdot měsíc + den}{366} \quad \textbf{(9)}$$

$$t_{den} = \frac{3600 \cdot hodina + 60 \cdot minuta + sekunda}{86400} \quad \textbf{(10)}$$

Tyto hodnoty jsou sice normalizované, ale mají jednu značnou nevýhodu – chceme-li z nich určit časovou vzdálenost např. denní doby pořízení fotografií, z nichž jedna byla pořízena sekundu před půlnocí a druhá sekundu po půlnoci, pouhý rozdíl hodnot t_{den} obou fotografií stačit nebude – výsledkem by byla hodnota blížící se v normalizovaném prostoru číslu 1, nicméně člověku je zřejmé, že správná hodnota by se měla blížit nule, protože rozdíl denní doby pořízení těchto fotografií jsou 2 sekundy.

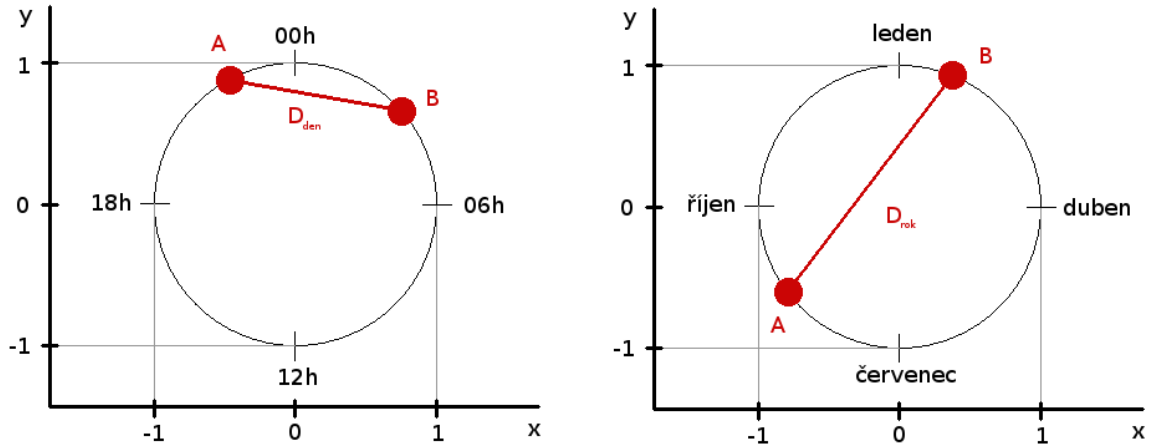
Tento nedostatek jsem vyřešil promítnutím těchto hodnot na kružnice, což je asi nejpřirozenější způsob řešení. Každá ze zmiňovaných hodnot t_{den} a t_{rok} bude reprezentována 2-rozměrnými souřadnicemi (celkem tedy 4 hodnoty), představující jejich pozici na jednotkové kružnici v rovině. Převod provádím pomocí následujících vztahů pro čas roku:

$$\begin{aligned} x_{rok} &= \sin(2\pi\, t_{rok}) \\ y_{rok} &= \cos(2\pi\, t_{rok}) \end{aligned} \quad \textbf{(11)}$$

Obdobně pro čas dne:

$$\begin{aligned} x_{den} &= \sin(2\pi\, t_{den}) \\ y_{den} &= \cos(2\pi\, t_{den}) \end{aligned} \quad \textbf{(12)}$$

Při této reprezentaci je výpočet časové vzdálenosti např. v rámci času dne možno realizovat prostou Eukleidovskou vzdáleností ve 2-rozměrném prostoru. Schématické znázornění této reprezentace času je znázorněno na obrázku *3.1*.



a) *b)*

Obrázek 3.1 – Schématické znázornění cyklické reprezentace času: a) čas dne, b) čas roku. Červeně znázorněné 2 časy a způsob měření rozdílu mezi nimi.

Vzhledem k tomu, že hned v úvodu projekční metody t-SNE je potřeba vypočítat vzdálenosti mezi mnohodimenzionálními body, tedy příznakovými vektory, je potřeba vhodně zvolit způsob reprezentace vzdáleností mezi příznakovými vektory. Běžná Eukleidovská vzdálenost, kterou používají autoři metody, není vhodná pro hodnoty histogramů, které tvoří jádro (a 99% dat) mnou používaných příznakových vektorů. Naopak pro ostatní hodnoty příznakového vektoru je Eukleidovská vzdálenost vhodná, včetně hodnot určujících čas dne a roční dobu pořízení fotografií. Proto jsem při implementaci zvolil *vícejádrovou vzdálenost*, kdy vzdálenosti všech hodnot příznakových vektorů, kromě histogramů, jsou počítány pomocí klasické Eukleidovské vzdálenosti, a pro vzdálenost mezi hodnotami reprezentujícími histogram je použit tzv. χ -kvadrát, který je vhodným měřítkem podobnosti/odlišnosti histogramů, kdy vzdálenost (odlišnost) histogramů určíme podle vztahu

$$d(\vec{a},\vec{b}) = \sum_i \frac{(a_i - b_i)^2}{a_i + b_i}, \qquad \textbf{(13)}$$

kde vektory $\vec{a}$ a $\vec{b}$ obshují hodnoty srovnávaných histogramů. Vzhledem k tomu, že histogramy i všechny ostatní části příznakových vektorů jsou normalizované, jak bylo uvedeno, stačí pro získání výsledné vzdálenosti dvou příznakových vektorů jen sečíst 4 samostatné vzdálenosti:

1) Euklidovská pro čas dne
2) Euklidovská pro čas roku
3) Euklidovská vzdálenost vektorů sdružujících hodnoty nastavení fotoaparátu (expozici, ISO a clonu)
4) χ -kvadrát histogramů.

Tyto 4 hodnoty můžeme dle libosti váhovat, abychom zvýraznili či zanedbali některou z těchto vlastností fotografie.

3.2 Prostá mapa s přibližováním

V této a následujících 2 částech popisuji možnosti použití projekce t-SNE pro zobrazení kolekcí fotografií, které jsem implementoval a otestoval.

V této části se zabývám možností, kterou jsem implementoval jako první, tedy prostou mapu s přibližováním, ve které jsou jednoduše všechny fotografie promítnuty do 2D mapy, která se pak uživateli přímo zobrazí a pracuje s ní. Při této první implementaci se na naší testovací kolekci fotografií osvědčila velká kvalita projekce t-SNE, při správném nastavení parametrů (zejména perplexita na co nejnižší hodnotu, použil jsem perplexitu o hodnotě *5*) se vytvořily jasně ohraničené kompaktní shluky podobných fotografií, viz obrázek *3.2*.

Na druhou stranu tento přístup se ukázal jako nepříliš vhodný v okamžiku, kdy uživatel mapu více oddálí. V takovém případě vyvstanou 2 problémy: zobrazuje se velké množství fotografií, které mohou zahltit výpočetní zdroje počítače a způsobit tak nepřípustně pomalé fungování systému, a k tomu se miniatury fotografií příliš překrývají a uživatel ztrácí přehled ve struktuře. Tyto problémy by mohly být vyřešeny výběrem vhodných reprezentativních fotografií, které by v případě překryvu byly přesunuty na popředí, a překryté fotografie na pozadí by mohly být ze zobrazení vypuštěny a zobrazeny až při takovém přiblížení, že nebudou reprezentativními fotografiemi překryty. V případě prosté mapy je však obtížné tyto reprezentativní fotografie určit. Navíc zobrazení prostou mapou nenabízí možnosti systematického procházení kolekce, což může způsobit špatnou orientaci uživatele v kolekci. Z těchto důvodů jsem se rozhodl dále implementovat zobrazení kolekce strukturálním způsobem a z vhodných struktur uvedených v kapitole 2 jsem zvolil statickou hierarchii sestavenou pomocí shlukování.

Obrázek 3.2 – Prostá mapa vzniklá projekcí t-SNE s perplexitou 5.

3.3 Hierarchický clustering

Jako další možnost zobrazení kolekce jsem implementoval statickou hierarchii pomocí aglomerativního clusteringu (shlukování), abych ze své implementace odstranil nedostatky prosté mapy.

Pomocí aglomerativního clusteringu v původním N-rozměrném prostoru příznakových vektorů vytvořím binární strom, který pak převedu na strom vyššího řádu (řád je volitelný, v mé implementaci jsem zvolil řád 16). V každém uzlu stromu pak seřadím všechny fotografie (listy, i nepřímé) každého přímého potomka-uzlu podle vzdálenosti od centroidu shluku, který onen přímý potomek-uzel reprezentuje.

Pro každý uzel promítnu všechny přímé potomky (tedy přímé potomky-listy a centroidy všech přímých potomků-uzlů) projekcí t-SNE – projekce se tedy provádí tolikrát, kolik je ve stromu uzlů,

nicméně každý uzel má v mém případě maximálně 16 přímých potomků, projekce tedy promítá jen 16 bodů, takže toto nemá negativní dopad na časovou náročnost.

Po provedení projekce můj algoritmus skládá spirálovitě kolem centroidů přímých potomků-uzlů všechny fotografie v jejich podstromu (všechny listy podstromu) seřazené podle vzdálenosti od centroidu shluku, který onen přímý potomek-uzel představuje, a skládá je tak aby se navzájem nepřekrývaly. Skládání běží, dokud nejsou spotřebovány všechny fotografie a dokud by další skládání nezpůsobilo příliš velké přiblížení reprezentantů 2 přímých potomků-uzlů. Tím docílí zobrazení maximálního možného množství reprezentativních fotografií všech podstromů a zároveň zamezí překryvu fotografií či splynutí skupin různých podstromů.

Při běhu aplikace pak uživatel vždy vidí jen přímé potomky-listy a reprezentanty přímých potomků-uzlů, kteří jsou od přímých potomků-listů barevně odlišeni. Kliknutím na fotografii-list uživatel fotografii vybere pro manipulaci, kliknutím na fotografii-reprezentanta uživatel přejde do příslušného uzlu, tedy do kořene příslušného podstromu. Takto uživatel může procházet strom od kořene až k listům, tedy jednotlivým fotografiím.

Problémem tohoto přístupu se ukázala volba způsobu řazení přednosti fotografií při výběru reprezentantů podstromu, centroid totiž nemusí být směrodatný – podstrom se například může skládat z 5 shluků, z nichž jeden je uprostřed a ostatní po stranách, a tím pádem by se mezi reprezentanty dostaly jen fotografie tohoto centrálního shluku, a uživatel před výběrem podstromu nebude vůbec vědět o tom, že se tam nachází i fotografie jiného typu.

Dalším problémem jsou nevhodně tvořené shluky, kdy je skupina podobných fotografií rozdělena mezi 2 sousední shluky skládající se z převážně jiných fotografií – tím se velmi podobné fotografie mohou dostat do zcela odlišných částí stromu, což je nežádoucí.

Oba tyto problémy společně způsobí, že se uživatel v kolekci špatně orientuje. Z toho vyplývá, že hierarchie shluků je pro tyto účely nevhodná, nicméně prostá mapa zmiňovaná v předchozí kapitole je rovněž nedostačující.

3.4 Možné úpravy t-SNE

V této části popisuji variantu, která se jeví jako nejvhodnější, tedy mapu s použitím časové osy (dále jen *Timeline*), kterou jsem implementoval.

Jedná se o kompromis mezi výše představenými možnostmi prosté mapy a strukturami – není zde žádná explicitní struktura, ale tím, že jsou fotografie řazeny podle času pořízení, je dosaženo určitého řádu a zlepšení orientace uživatele. Je vždy zobrazena jen část časové osy, tedy jen část fotografií, zobrazený úsek bude možno zvětšovat či zmenšovat a posouvat jej jedním nebo druhým směrem. Fotografie jsou podle času zobrazovány zleva doprava, nad nimi je časová osa zobrazující data a časy aktuálně zobrazeného úseku fotografií a vpravo pak bude umístěna část rozhraní pro přiřazováni tagů vybrané fotografii či skupině fotografií. Náhled prototypu zobrazení kolekce, zatím bez posouvání, změny měřítka a bez části pro přiřazování tagů, je na obrázku *3.3*.

Jedná se o úpravu projekce t-SNE, inicializace průmětů je náhodná jen v ose Y, na ose X jsou průměty inicializovány postupně v závislosti na času pořízení. Při iteracích t-SNE je pak pohyb průmětů po ose X omezen tak, aby byla do jisté míry zachována časová posloupnost, ale zároveň ne moc, aby se podobné fotografie pořízené v přibližně stejnou dobu mohly shlukovat.

Nabízí se několik způsobů jak pohyb po ose X omezit:

1. Použít t-SNE na projekci do 1-rozměrného prostoru a pohybovat fotografiemi jen po ose Y, souřadnice X nechat tak, jak se inicializují
2. Přiřadit velkou váhu časovým údajům při počítání vzdálenosti v původním vícerozměrném prostoru
3. Cílem je aby s každou fotografií reagovaly jen fotografie v určitém časovém okruhu kolem ní, toto by se mohlo zohlednit při počítání podobností – nastavit podobnost na 0 pro všechny fotografie, kromě těch pořízených v podobnou dobu.

4. Vytvořit pro každou fotografii *kotevní bod* ve 2-rozměrném prostoru, a omezovat pohyb průmětu podle jeho vzdálenosti od kotevního bodu
5. Utvořit lokální struktury mezi sousedními průměty a s určitou tolerancí udržovat poměr vzdáleností mezi těmito průměty
6. Zjišťovat velikosti studentových distribucí podobně, jako směrodatnou odchylku u Gaussovských distribucí ve vícerozměrném prostoru, a pomocí vhodného nastavení jejich velikosti omezit počet průmětů, které jsou brány v úvahu, a tím zamezit přitažlivým silám mezi časově vzdálenějšími fotografiemi
7. Spojení gradientů několika KL-divergencí tak, aby byl pohyb průmětů po ose X omezen

Tyto varianty projekce implementuji a postupně otestuji vhodnost jednotlivých variant jak z hlediska výpočetní náročnosti, tak z hlediska vizuálních výsledků a s nimi spojené použitelnosti uživatelem a efektivity práce s kolekcí a tagování v důsledku.

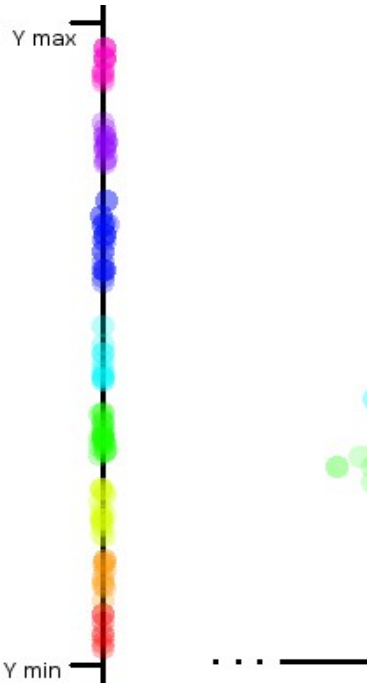
Obrázek 3.3 – první prototyp zobrazení kolekce fotografií pomocí Timeline (časové osy). Nahoře časová osa, fotografie jsou řazené podle času zleva doprava a tvoří shluky a zároveň je použita jistá úroveň zamezení překryvu mezi fotografiemi.

3.5 Testované úpravy t-SNE

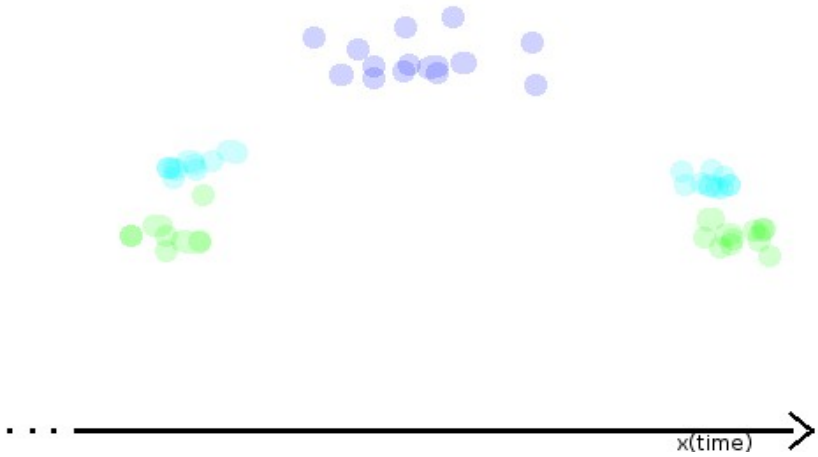
Výběr vhodných úprav z variant, které jsem uvedl v předchozí kapitole, jsem prováděl od nejjednodušších a zároveň výpočetně nejméně náročných variant, tedy tak, jak jsou v textu číslovány.

***Varianta 1* – použití t-SNE projekce jen do 1-rozměrného prostoru**, tak, aby se fotografie pohybovaly jen ve směru osy Y, a ve směru osy X zůstaly na původních pozicích, inicializovaných tak, aby byly fotografie zleva doprava řazeny podle času pořízení – se nakonec neukázala jako příliš vhodná. Bez jakékoliv volnosti pohybu ve směru osy X se minimalizuje možnost utvoření shluku podobných fotografií, které jsou ovšem hlavním přínosem projekce. Nicméně kdybych tuto variantu použil, určitě by bylo potřeba omezit množství sousedních fotografií (třeba na 10 okolních fotografií), ovlivňujících gradient průmětu pozice fotografie v 2-rozměrném prostoru a potažmo jeho výslednou pozici *(lokalizovaná 1-rozměrná projekce)*. Toto zaprvé podstatným způsobem zrychlí přípravnou i iterativní fázi projekce, a zadruhé podpoří vnitřní hustotu jednotlivých clusterů i jejich vzájemnou separaci – nedojde k ovlivňování pozice fotografie *A* fotografiemi, které jsou od *A* časově velmi vzdálené. Kdyby tomuto vlivu nebylo zabráněno, tak by t-SNE projekce do 1-rozměrného prostoru

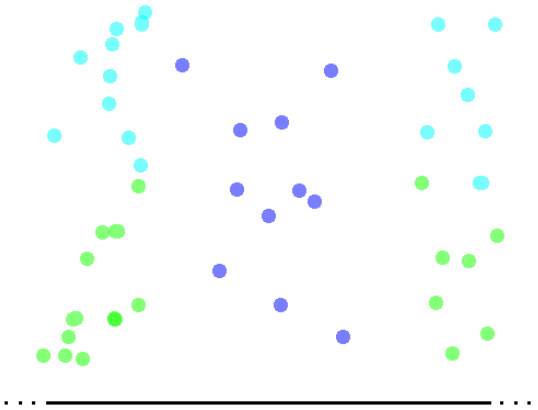
správně vytvořila průmět, znázorněný na obrázku *3.4*, jenže naše fotografie by nebyly zobrazeny všechny se stejnou X-ovou souřadnicí, ale do značné šíře podle času pořízení, a došlo by zmenšení rozlišitelnosti shluků (obrázek *3.5*), nepomohla by ani lokální normalizace, ta by pouze k malé vzdálenosti mezi shluky přidala vnitřní zřídnutí jednotlivých shluků v ose Y (obrázek *3.6*). Naopak při omezení množství použitých sousedních fotografií dochází k projekci jen omezeného množství fotografií, například pro první třetinu časové osy X použitého příkladu budou vůči sobě uspořádány jen fotografie zeleného a tyrkysového shluku, které se tím pádem zřetelně oddělí, jak je ilustrováno na obrázku *3.7*.



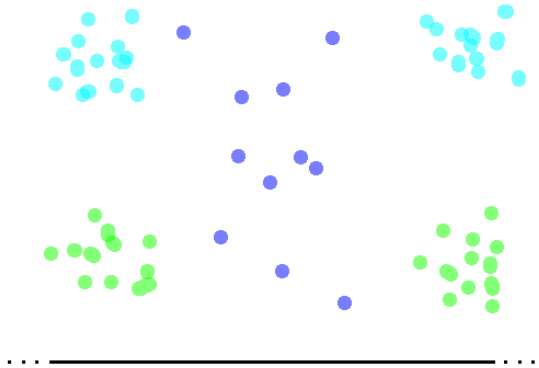
Obrázek 3.4
globální 1D projekce



Obrázek 3.5 – globální 1D projekce roztažená podle času, malá rozlišitelnost zelených a tyrkysových shluků



Obrázek 3.6 – globální 1D projekce, roztažená podle času, přidaná lokální normalizace – stále špatná rozlišitelnost, navíc snížená vnitřní hustota shluků.



Obrázek 3.7 – lokalizovaná 1D projekce s přidanou lokální normalizací – husté a jasně oddělené zelené a tyrkysové shluky.

Varianta 2 – přiřazení velké váhy času pořízení fotografie při počítání vzdáleností v původním vícerozměrném prostoru. V tomto případě je tedy činitelem, zajišťujícím udržení časové posloupnosti přiřazení vysoké váhy času pořízení, konkrétně tak vysoké, aby se fotografie pořízené ve vzájemně časově blízkých okamžicích vzájemně přitahovaly více, než s fotografiemi časově vzdálenými, a to i když jsou si s časově vzdálenými fotografiemi podobnější vizuálními aspekty (barevné histogramy) či některým z jiných rozměrů příznakového vektoru. Zároveň je však potřeba nenastavit váhu času tak vysokou, aby úplně vyrušila vliv ostatních rozměrů příznakových vektorů a tím vlastně vyrušila přínos projekční metody. Samozřejmě neexistuje jen jedno univerzálně nejlepší nastavení vah času a ostatních rozměrů příznakových vektorů, je zde tedy jistá variabilita v závislosti na požadavcích uživatele na míru uspořádání fotografií podle času pořízení a na míru jejich seskupení podle ostatních aspektů. Mezi těmito mírami existuje vztah nepřímé úměrnosti, nicméně každé nastavení, které bude shlukovat vizuálně podobné fotografie, způsobí do jisté míry lokální zpřeházení fotografií co se týče času pořízení. To, jak moc lokální zpřeházení, tedy o kolik míst se může fotografie posunout mimo své časové pořadí, lze ovlivnit právě nastavenou velikostí váhy času a ostatních rozměrů příznakového vektoru.

Pro tento účel však není možné použít cyklickou reprezentaci času, jak byl popsán v kapitole *3.1*, který v sobě neobsahuje informaci o ničem jiném, než denní a roční době, ale je potřeba použít nějakou formu absolutního času. Pro porovnání dvou absolutních časů jsem do vzdálenosti dvou příznakových vektorů přičetl hodnotu vypočítanou podle následujícího vztahů:

$$seconds_{\Delta} = 32140800 \cdot y_{\Delta} + 2678400 \cdot mon_{\Delta} + 86400 \cdot d_{\Delta} + 3600 \cdot h_{\Delta} + 60 \cdot min_{\Delta} + sec_{\Delta} \quad \textbf{(14)}$$

$$T_{\text{abs}} = \frac{100000}{32140800} \cdot seconds_{\Delta} \quad \textbf{(15)}$$

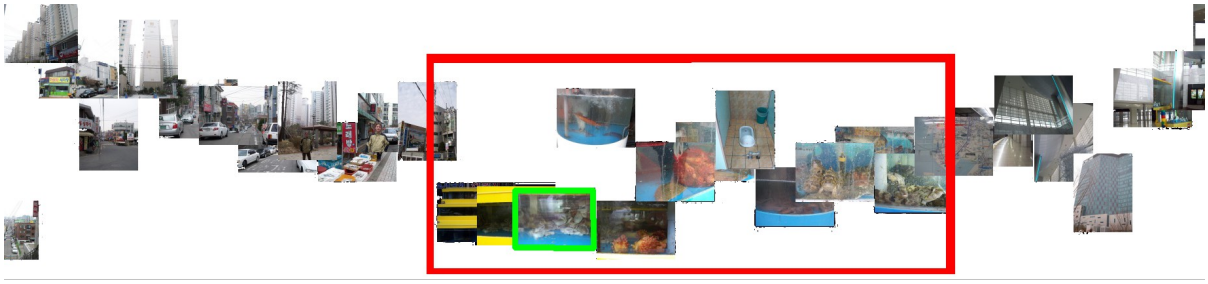
$$D = W_h \cdot \chi^2_{dist}(H_A, H_B) + W_{day} \cdot E_{dist}(C_{day\,A}, C_{day\,B}) + W_{year} \cdot E_{dist}(C_{year\,A}, C_{year\,B}) + W_{cam} \cdot E_{dist}(Cam_A, Cam_B) + W_{\text{abs}} \cdot T_{\text{abs}} \quad \textbf{(16)}$$

Vzorec (14) představuje výpočet rozdílu ve vteřinách mezi dvěma danými časy, kde proměnné na pravé straně vzorce představují po řadě rozdíl v roce, měsíci, dni, hodině, minutě a sekundě porovnávaných časů. Vzorec (15) převádí převod z vteřin na roky (dělitel ve zlomku je počet vteřin roku) a následné zvětšení hodnoty na sto-tisícinásobek – v praxi zajistí to, že při vstupní váze s hodnotou *1* pro absolutní čas a hodnotou *1* pro barevné histogramy bude váha absolutního času taková, aby vhodným způsobem zajistila minimální přeházení fotografií z hlediska času, a zároveň dala prostor pro shlukování časově blízkých fotografií. Hodnota 100 000 byla zjištěna experimentálně a souvisí s charakterem použitých příznakových vektorů a způsobem počítání vzdálenosti pro ostatní dimenze příznakových vektorů – při použití jiných příznakových vektorů nebo při jiném způsobu počítání vzdáleností mezi nimi se tato hodnota může lišit a bude potřeba ji opět experimentálně stanovit.

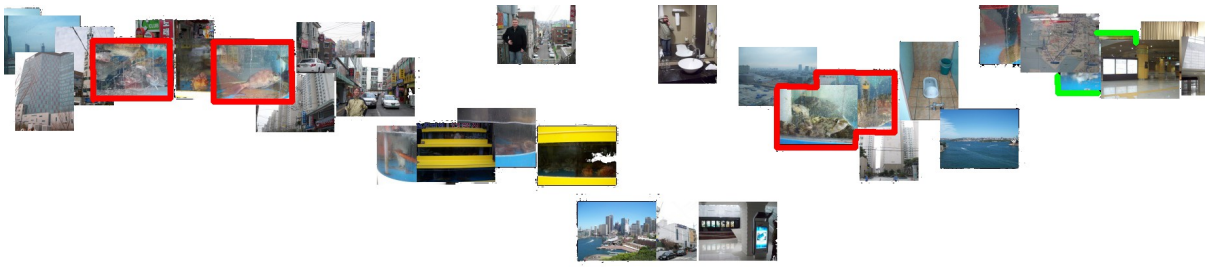
Vzorec (16) pak představuje celkový výpočet vzdálenosti mezi dvěma fotografiemi (resp. příznakovými vektory), kde $W_{něco}$ představuje vždy externě nastavitelnou váhu, přiřazenou dané částečné vzdálenosti, $E_{dist}(...)$ představuje klasickou Eulerovskou vzdálenost argumentů, $\chi^2_{dist}(...)$ představuje χ -kvadrát argumentů, H představují histogramy porovnávaných fotografií, C_{day}, C_{year} představují po řadě cyklickou reprezentaci času dne a roku, a hodnoty *Cam* představují hodnoty nastavení fotoaparátu v okamžiku pořízení fotografie. Používám pro výpočet vzdálenosti tedy i cyklický i absolutní čas, ovšem absolutní s činitelem 100 000.

Nevýhodou této varianty je neoptimálnost – zbytečně se ztrácí výpočetní čas, protože jsou počítány vlivy úplně všech fotografií, i těch, které by vzhledem ke své časové vzdálenosti neměly být brány v úvahu z hlediska vizuálních aspektů fotografií – neovlivňují vzdálené fotografie co se týče

vizuálních aspektů (směr osy *Y* a částečně osy *X*), pouze jsou potřebné pro příspěvek k udržení vzdálených fotografií na svém místě co se týče časového uspořádání (ve směru osy *X*). V tomto případě však nelze použít lokalizovanou projekci tak jako u varianty číslo *1*, nebo alespoň ne ve stejné podobě. Při lokalizaci projekce u varianty *2* totiž dochází k tomu, že pokud nastavíme, že pozice každé fotografie je dána pouze vzdálenostmi v původním prostoru příznakových vektorů k *N* časově nejbližším fotografiím (*N/2* pořízených dříve a *N/2* později), pak vzhledem k pohyblivosti fotografií i ve směru osy *X* dojde k tomu, že se *N* fotografií umístí správně vůči sobě, ale často dochází k roztažení těchto *N* fotografií ve směru osy *X* tak, že se promíchají s dalšími fotografiemi mimo těchto *N* fotografií. Je to pravděpodobně dáno tím, že fotografie nejsou dostatečně odpuzovány okolními fotografiemi, které jsou pořízeny ve velmi odlišném čase, protože se lokalizací zanedbávají, a tudíž se jejich projekce roztáhne do šířky, jak je znázorněno na obrázcích *3.8, 3.9* a *3.10*. Lokalizaci jsem tedy ve variantě s velkou váhou na čas nepoužil – použití by vyžadovalo přidání nějakého dalšího prostředku pro omezení pohybu po časové ose *X*.



Obrázek 3.8 – Obyčejná varianta s velkou váhou na čas, při níž se berou v potaz vztahy mezi všemi fotografiemi. Povšimněte si červeně označené skupiny fotografií akvária (mimo výjimky) a speciálně zelené fotografie – v této projekci jsou seřazeny časově i seskupeny vizuálně tak, jak mají, a jsou pohromadě.



Obrázek 3.9 – Lokalizovaná varianta – bere se v úvahu jen 20 okolních fotografií (10 před a 10 po pořízení dané fotografie) – fotografie akvária se promíchaly s ostatními, které jsou zaprvé pořízeny v jinou dobu, a zadruhé mají jiné vizuální aspekty.

Obrázek 3.10 – Lokalizovaná varianta, kde se bere v úvahu jen 6 okolních fotografií (3 před a 3 po pořízení dané fotografie) – fotografie akvária se rozutekly ještě mnohem dál. Mají být časově někde na středu, ale například zelená fotografie je téměř na kraji.

I přesto jsem se rozhodl tuto metodu použít, protože výsledná projekce je velice kvalitní a právě díky váhování snadno ovlivnitelná a plynule nastavitelná mezi prvním extrémem *jen podle času* a druhým extrémem *jen podle vizuálních vlastností*.

Dále jsem k metodě přidal v předzpracování i v následném zpracování po dokončení projekce prahování maximálního kroku mezi fotografiemi ve směru časové osy, osy *X*, aby nedocházelo k vzniku příliš velkých mezer mezi fotografiemi při použití ve výsledném uživatelském rozhraní.

Po projekci pak ještě provádím klouzavou normalizaci *Y*-ové souřadnice, aby nedocházelo ke kumulaci fotografií do úzkého pásu, kde by byly ve výsledném rozhraní špatně rozlišitelné a ztěžovalo by to procházení a výběr požadované skupiny fotografií. Klouzavou normalizaci provádím vždy normalizací *Y*-ové souřadnice fotografie podle minima a maxima z *N* sousedních fotografií (běžná hodnota *N* je *10*).

3.6 Implementace uživatelského rozhraní

Uživatelské rozhraní jsem se rozhodl implementovat jako online webovou aplikaci (typu *Flickr*), ale samozřejmě doposud uváděné principy je vhodné použít i při implementaci běžné „offline“ aplikace. V této kapitole stručně popíšu vytvořené aplikace, které budou součástí výsledného rozhraní.

Jedná se o sadu na sebe navazujících aplikací, které tvoří zřetězenou linku. První aplikace, implementovaná v C++ za použití knihovny *libexif* (pro práci s EXIF informacemi fotografií) a knihovny *OpenCV* (pro práci s obrazovými daty fotografií), připraví příznakové vektory zpracovávaných fotografií – zjistí EXIF informace, převede časové údaje do cyklické podoby, přidá histogramy, které jsou vypočtené způsobem popsaným ve článku [10] pomocí aplikace převzaté od Ing. Michala Hradiše, a vše normalizuje, jak bylo popsáno v kapitole *3.1*. Výstup předá aplikaci implementované v jazyce C++, která provede projekci příznakových vektorů podle zadaných parametrů pomocí upravené metody t-SNE, jejíž původní varianta je popsaná v kapitole *2.5* a použité úpravy v druhé části kapitoly *3.5*.

Výsledek projekce je předán webovému rozhraní, které je založené na frameworku *Django* (webový framework v jazyce Python). Server si uloží cesty k souborům fotografií a souřadnice jejich projekcí do databáze a nástroj pro přípravu příznakových vektorů a pr projekci už není nad kolekcí fotografií vícekrát potřeba používat.

Klientskou část webového rozhraní jsem implementoval pomocí HTML, JavaScriptu a JavaScriptové knihovny *KineticJS*, což je knihovna pro snazší práci s plátnem (canvas) v HTML5. Pomocí knihovny *KineticJS* na plátnu zobrazuji celé uživatelské rozhraní.

Mimo zmiňované aplikace jsem implementoval také aplikaci v C++ s použitím knihovny *OpenCV* pro vyrenderování projekce fotografií do obrázku, pro snazší testování modifikací projekce.

Implementaci uživatelského rozhraní detailněji popíši v následujících podkapitolách.

3.6.1 Server

Jak bylo řečeno, na serveru mé webové aplikace se pomocí zřetězené linky 3 aplikací (tvorba histogramů pomocí programu převzatého z [10], příprava příznakových vektorů, projekce) provede celé zpracování sady fotografií a projekce z prostoru příznakových vektorů do 2-rozměrného prostoru obrazovky pomocí *Timeline* projekce popsané v kapitole *3.4* a *3.5*. Výstup projekce se předá serveru v textové podobě, kde formát každého řádku je následující:

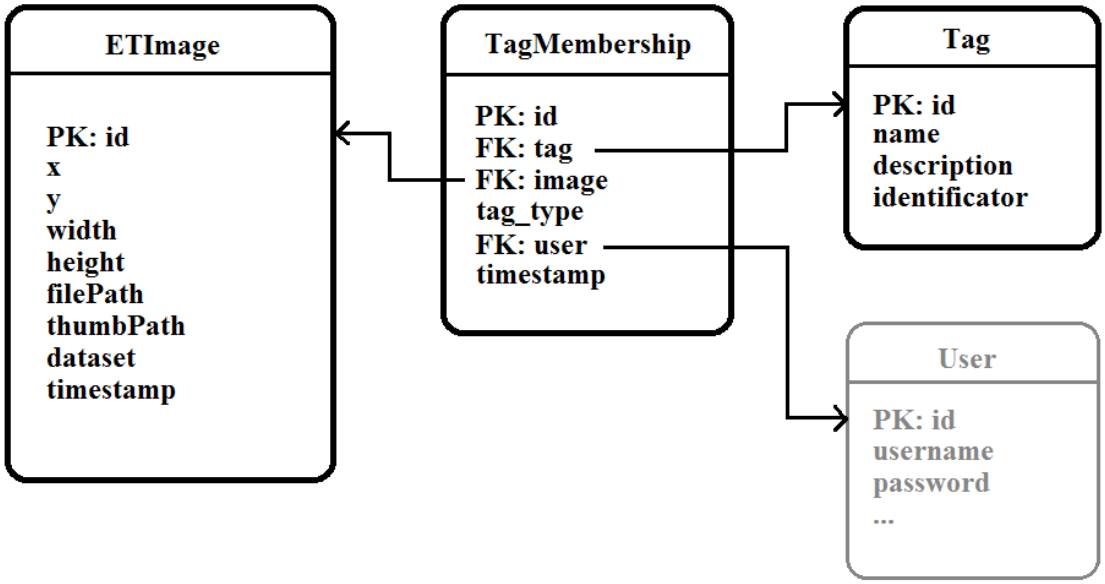
<PATH> <W> <H> <X> <Y> <YYYY>:<MM>:<DD> <HH>:<MM>:<SS>

kde *<PATH>* představuje cestu k souboru fotografie, *<W>* a *<H>* jsou výška a šířka fotografie normalizované do intervalu <0; 1>, *<X>* a *<Y>* jsou souřadnice projekce fotografie do 2-rozměrného

prostoru, normalizované do intervalu <0; 1>, *<YYYY>:<MM>:<DD>* představuje datum pořízení fotografie a *<HH>:<MM>:<SS>* čas pořízení fotografie.

Zmíněné 3 aplikace zřetězené linky na serveru implementují algoritmy detailně popsané v předchozích kapitolách a ve článku [10], v této kapitole se blíže zaměřím na implementaci samotného webového serveru, který je implementován pomocí frameworku *Django*, používajícího programovací jazyk *Python*.

Databáze webového serveru je vytvořena pomocí databázového systému *SQLite 3*, ve kterém vytvářím 3 své vlastní tabulky a používám 1 tabulku pro správu a přihlašování uživatelů, implicitně vytvořenou frameworkem *Django*. Schéma použité databáze je znázorněno na obrázku *3.11*.



Obrázek 3.11 – schéma databáze prototypu webového serveru.

Tabulka ***ETImage*** uchovává informace o fotografiích – normalizované souřadnice projekce, normalizovanou šířku a výšku fotografie, cestu k souboru fotografie, cestu k souboru zmenšeniny fotografie, název datasetu (kolekce fotografií) a časový údaj pořízení fotografie.

Tabulka ***Tag*** uchovává všechny známé tagy.

Tabulka ***User*** je implicitně tvořena frameworkem *Django* pro uchování seznamu uživatelů a jejich přihlašování.

Tabulka ***TagMembership*** uchovává příslušnost tagu z tabulky *Tag* k fotografii z tabulky *ETImage*. Parametr *tag_type* může nabývat hodnot *„positive“* nebo *„negative“*, který je zde připraven pro potřeby pozitivního i negativního tagování, kdy pomocí tagů neurčujeme jen co se na fotografii nachází, ale můžeme explicitně říci i co se na fotografii nenachází (využitelné pro tagování pro strojové učení klasifikátorů fotografií). Cizí klíč *user* odkazující na záznam v tabulce uživatelů je zde z důvodů experimentu, který je popsán dále v textu.

Volání webového serveru

Má implementace používá otevřenou množinu tagů, tedy uživatel není omezen nějakou předem stanovenou množinou přípustných tagů, ale může přiřazovat libovolně pojmenovaní tagy. Z tohoto důvodu nejsou při zasílání informací o příslušnosti tagů nebo při voláních pro přiřazení nebo odebrání tagu daným fotografiím zasílány identifikátory tagů *Tag.id*, ale místo toho přímo jejich názvy *Tag.name*. Tento způsob je sice redundantní co se týče objemu posílaných dat při těchto operacích,

ale na druhou stranu použití prvního způsobu, tedy zasílání *Tag.id*, by pro otevřenou množinu tagů znamenalo podstatné zvýšení síťového provozu potřebného pro uchovávání aktuálnosti tabulky přiřazující již existujícím názvům tagů jejich *Tag.id*, nehledě na potřebu větší paměti na straně klienta pro uchování této tabulky. Z tohoto důvodu se jeví vhodnější a efektivnější zasílat přímo *Tag.name*, přičemž server snadno zjistí, jestli tag z daným názvem už v tabulce *Tag* existuje, nebo je potřeba jej tam vytvořit. Server s názvy tagů zachází bez citlivosti na velikost písmen, názvy tagů *Tag.name* jsou uchovávány v malých písmenech.

Následuje seznam významných volání (vynechána jsou triviální volání typu přihlášení a odhlášení uživatele a podobně):

1) **load_dataset - načtení datasetu (= kolekce fotografií)** – načtení a zpracování souboru, který je výstupem zřetězené linky 3 programů uvedených v úvodu této kapitoly a který obsahuje informace o projekci kolekce fotografií ve výše uvedeném formátu. Přidá záznamy do tabulky *ETImage*, přičemž název kolekce fotografií, který se do tabulky vkládá do sloupce *dataset*, je parametrem tohoto volání.

2) **view – zobrazení datasetu (= kolekce fotografií) –** zobrazí kolekci fotografií pomocí vytvořeného uživatelského rozhraní. Toto volání pouze klientovi pošle HTML5 šablonu používající JavaScriptový kód, jehož činnost je blíže popsána v podkapitole*3.6.2*.

3) **get_timeline – načtení informací –** pošle klientovi JSON strukturu obsahující informace o kolekci fotografií. JSON struktura obsahuje záznamy z tabulky *ETImage*, které mají ve sloupci *dataset* název kolekce fotografií, který je parametrem tohoto volání.

4) **get_tags – získání tagů, které byly přiřazeny specifikovaným fotografiím** – vstupním parametrem je JSON objekt obsahující seznam primárních klíčů *ETImage.id* fotografií, pro které chceme získat seznamy přiřazených tagů. Volání pošle klientovi JSON strukturu v následujícím formátu:

```
{
        "<ETImage1.id>" : [<Tag1.name>, <Tag2.name>, <Tag3.name>, …],
        "<ETImage2.id>" : [<Tag1.name>, <Tag5.name>, <Tag7.name>, …],
        ...
}
```

5) **add_tag – přiřazení tagu specifikovaným fotografiím –** vstupním parametrem tohoto volání je JSON objekt obsahující název přiřazovaného tagu, seznam *ETImage.id* fotografií, kterým tag chceme přiřadit, a typ tagu (pozitivní nebo negativní, pro případné budoucí použití pro negativní tagování), ve formátu:

```
{
        "tag" : "tag_name",
        "photo_ids" : [<ETImage1.id>, <ETImage2.id>, …],
        "type" : <type_value>
}
```

Volání vrací informaci o úspěšnosti operace.

6) **remove_tag – odebere tag specifikovaným fotografiím –** vstupním parametrem tohoto volání je JSON objekt obsahující název odebíraného tagu a seznam *ETImage.id* fotografií, kterým tag chceme odebrat, ve formátu:

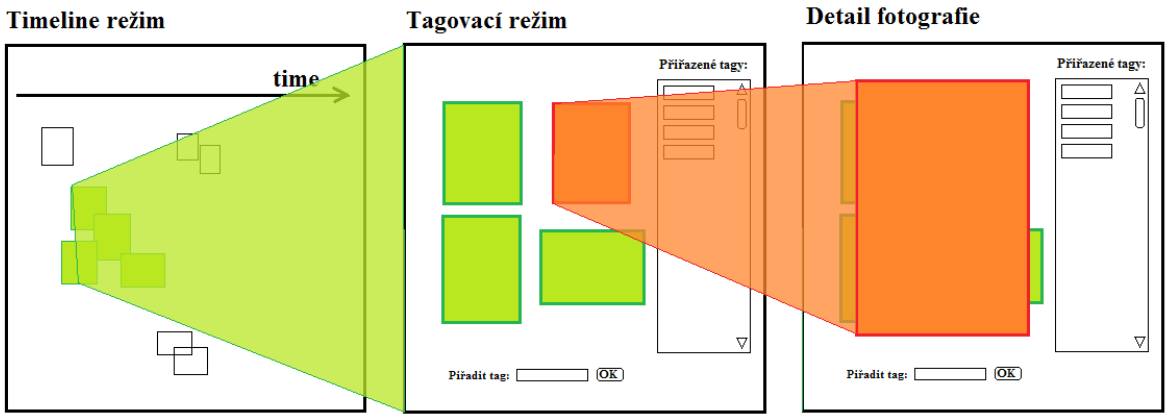
{
“tag” : “tag_name”,
“photo_ids” : [<ETImage1.id>, <ETImage2.id>, …]
}

Volání vrací informaci o úspěšnosti operace.

3.6.2 Klient

Samotné grafické uživatelské rozhraní jsem implementoval na klientovi pomocí JavaScriptu, HTML5 prvku *canvas* a open-source knihovny *KineticJS*, které práci s HTML5 prvkem *canvas* do jisté míry zapouzdřuje. Jak už bylo řečeno v předchozí podkapitole, webový server na volání pro zobrazení projekce datasetu *(view)* odpoví pouze HTML5 šablonou odkazující se na JavaScriptový skript, který provede zbytek potřebných operací. Tento skript je tedy jádrem klientské části mého rozhraní.

Uživatelské rozhraní má dvě části – první část, *Timeline režim*, obsahuje zobrazenou projekci kolekce fotografií, reprezentovaných zmenšenými náhledy, s horizontální časovou osou, kde může uživatel prohlížet a efektivně vybírat jednu či více fotografií zároveň pro tagování. Poté, co uživatel vybere fotografie, kterým chce přiřadit tagy, přepne uživatelské rozhraní do jeho druhé části, *Tagovacího režimu. Tagovací režim* slouží k efektivnímu přiřazování tagů fotografiím, kde jsou aktuálně vybrané zvětšené fotografie v plném rozlišení zobrazeny v mřížce a lze přiřazovat tagy všem těmto fotografiím zároveň nebo jejich libovolné podmnožině. Dané fotografie lze po jedné zvětšit přes celou obrazovku, aby je mohl uživatel prohlížet detailněji. Poté, co uživatel dokončí práci s aktuálně vybranou skupinou fotografií, přepne uživatelské rozhraní zpět do *Timeline režimu*, kde může vybírat další fotografie pro další tagování. Schéma režimů je zobrazené na obrázku *3.12*.

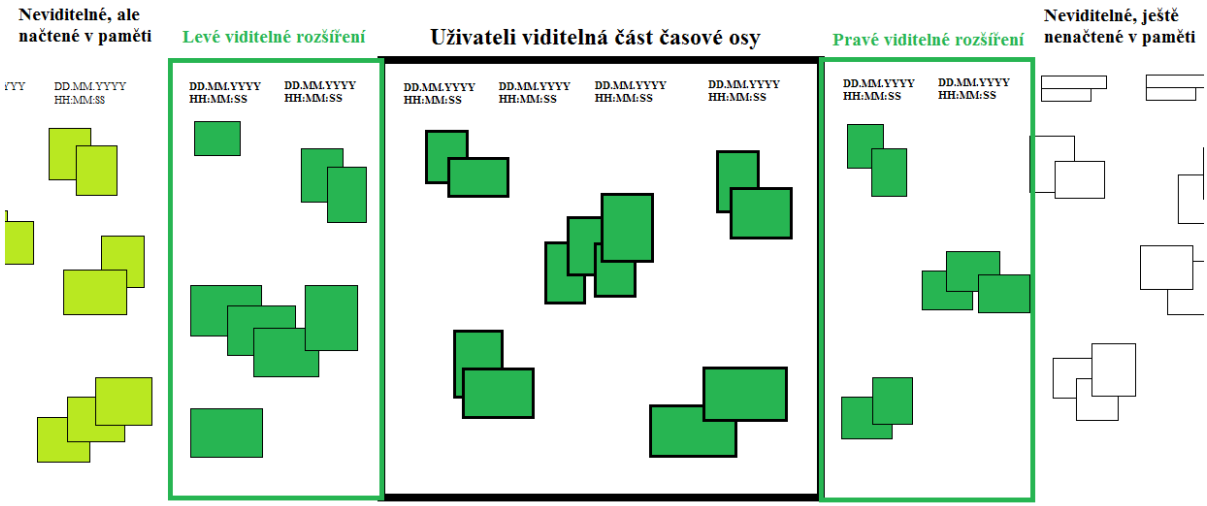


Obrázek 3.12 – schéma režimů uživatelského rozhraní. V Timeline režimu uživatel prohlíží a vybírá fotografie, se kterými chce pracovat, v Tagovacím režimu může libovolné podmnožině množiny vybraných fotografií přiřazovat nebo odebírat tagy, případně pro detailnější prohlížení jednotlivých fotografií může fotografii zvětšit přes celou obrazovku (Detail fotografie).

Javascriptový skript, který je jádrem klientské části mého uživatelského rozhraní, nejprve provede potřebnou inicializaci všech interních proměnných a inicializuje pomocí knihovny *KineticJS* prvek *canvas,* vrstvy zobrazovací struktury používané v *KineticJS (stage)* a všechny zobrazované statické prvky – tlačítka, popisky, barevný přechod na pozadí a podobně.

Dále zavolá volání serveru pro načtení informací požadované kolekce fotografií *(get_timeline)*, kterým získá informace o fotografiích. Následuje načtení a inicializace aktuálně zobrazených a nejbližších okolních nezobrazených fotografií a popisů na časové ose. Ostatní fotografie a popisy se zatím neinicializují, aby načtení na začátku práce netrvalo příliš dlouho. Zbylé fotografie a popisy

časové osy se načítají dynamicky podle posunu časové osy uživatelem při projíždění kolekcí. Procházení kolekcí je realizováno posunem celé jedné zobrazovací vrstvy zobrazovacího objektu knihovny KineticJS. Tuto vrstvu jsem vyhradil pouze pro fotografie a popisy časové osy. Při posunu této vrstvy vždy po určitém kroku posunu aktualizuji nastavení fotografií a popisů tak, aby se fotografie a popisy mimo zobrazenou část i její nejbližší okolí vůbec nezobrazovaly a nedocházelo ke zbytečné zátěži. Schéma načítání, zobrazování a skrývání fotografií a popisů je znázorněno na obrázku *3.13*. Další činnost skriptu už spočívá jen v reakcích na události spouštěné prací uživatele.



Obrázek 3.13 – Schéma načítání, zobrazování a skrývání fotografií a popisů časové osy. Vyplněné obdélníky představují již načtené fotografie, světlejší výplň znamená, že je fotografie načtená, ale není aktuálně brána v potaz při zobrazování (je skrytá – i z hlediska zobrazování v knihovně KineticJS). Prázdné obdélníky představují fotografie, které ještě nebyly načteny ze serveru. Popisy nahoře představují popisy časové osy, tučný font znamená, že jsou brány v potaz při zobrazování, obyčejný font že brány v potaz nejsou, ale jsou načteny, prázdné obdélníky znamenají, že ještě nebyly načteny do paměti. Předpokládejme, že uživatel začal pracovat v nějakém místě, a pak se posunul na časové ose doprava.

Ovládání a popis uživatelského rozhraní

Timeline režim:

1) najetím myší na kraj obrazovky nebo stisknutím příslušné šipky na klávesnici se posune aktuálně zobrazená část časové osy

2) kolečkem myši se přibližuje/oddaluje náhled – z hlediska časové osy. Náhledy fotografií zůstanou stejně velké, jen se mění jejich hustota

3) kliknutím/tažením levým tlačítkem myši se přidají fotografie do výběru

4) kliknutím/tažením pravým tlačítkem myši se vyberou fotografie a rovnou zvětší do tagovacího režimu

5) kliknutím pravým do volného prostoru se zruší aktuální výběr

6) mezerníkem nebo enterem se zvětší vybrané fotografie do tagovacího režimu

Tagovací režim:

1) kliknutím na fotografii levým se fotografie aktivuje/deaktivuje

2) vpravo je seznam tagů, které jsou přiřazené alespoň jedné aktivní fotografii

3) najetím na tag se zvýrazní všechny fotografie, které ho mají přiřazené

4) najetím na fotografii se zvýrazní všechny tagy, které má fotografie přiřazené

5) jakýkoliv text napsaný do vstupního řádku dole se stiskem tlačítka nebo stiskem Enteru na klávesnici přiřadí jako tag všem aktivním fotografiím

6) kliknutím na tag v seznamu přiřazených tagů odeberete tag všem aktivním fotografiím

7) levý klik na fotografii + CTRL = aktivuje tuto fotografii a deaktivuje všechny ostatní

8) dvojitý levý klik na fotografii = zvětší danou fotografii přes celé (režim *Detailu fotografie*), aktivní je jen tato fotografie, po kliknutí na ni nebo někam do prostoru se zase všechno vrátí zpátky do původní aktivace/deaktivace

9) klik na fotografii pravým – úplně odstraní fotografii z výběru, aby ostatní fotografie mohly třeba být větší

10) klik na fotografii levým + SHIFT = aktivuje všechny fotografie od té, na kterou bylo kliknuto minule, po řádcích až po tu, na kterou bylo kliknuto teď

11) klik na fotografii levým + SHIFT + CTRL = totéž co 10) ale k tomu deaktivuje všechny ostatní fotografie

12) DELETE odstraní z výběru všechny neaktivní fotografie

13) kliknutí na pozadí nebo zmáčknutí ESC vrátí uživatele zpět do *Timeline režimu*.

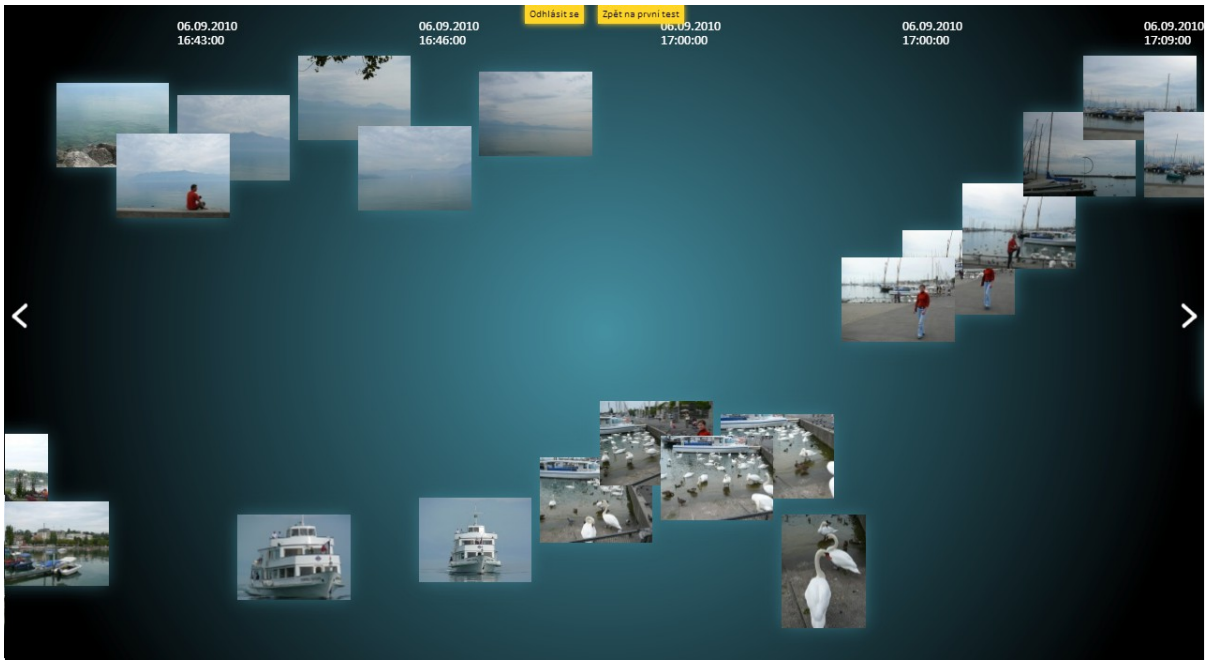
Komunikace se serverem při tagování

Při přechodu z *Timeline režimu* do *Tagovacího režimu* je odeslán požadavek na server zaprvé o fotografie v plné velikosti (v *Timeline režimu* jsou jen zmenšené náhledy) a zadruhé o tagy přiřazené fotografiím ve výběru (volání *get_tags*, viz podkapitola *3.6.1*) a přijatou strukturu si skript uloží do paměti. Podle ní zjišťuje, které všechny tagy jsou v daném výběru zastoupeny a který tag je přiřazen které konkrétní fotografii.

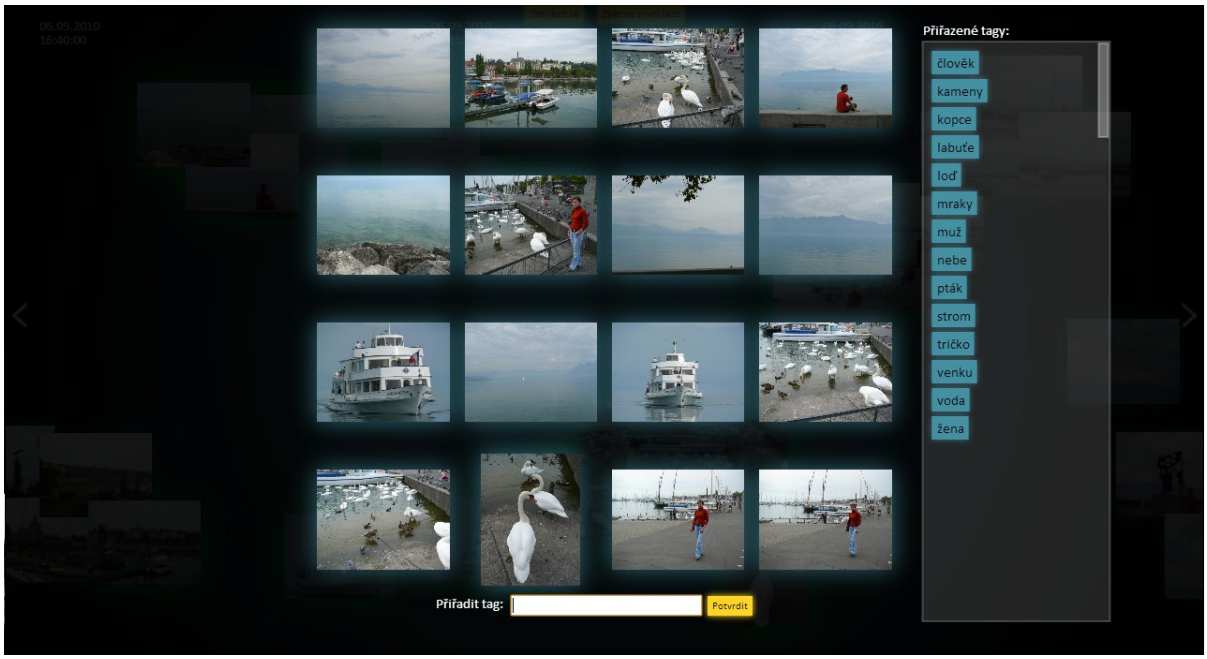
Když uživatel přidává nebo odebírá tagy, tak je na server odeslán vždy jen požadavek o přiřazení nebo odebrání tagu (volání *add_tag* nebo *remove_tag*) a nedochází už k dalšímu zasílání požadavku *get_tags*, pouze k aktualizaci lokálně uložené datové struktury, aby nedocházelo ke zbytečnému zvyšování síťového provozu. Toto opatření lze v mém případě použít, protože každý uživatel může zobrazovat nebo mazat pouze tagy, které sám přidal. Jak je patrné ze schématu v obrázku *3.11*, *TagMembership* obsahuje cizí klíč *user* odkazující na uživatele, který jako jediný s daným záznamem v tabulce *TagMembership* může manipulovat. V případě sdíleného přístupu k tagování více uživateli by bylo toto potřeba nějak ošetřit, aby byla synchronizována databáze na serveru a obsah viditelný klientovi, nejspíše by bylo potřeba zasílat požadavek *get_tags* při každém přidání i odebrání tagu.

Snímky vytvořeného prototypu uživatelského rozhraní

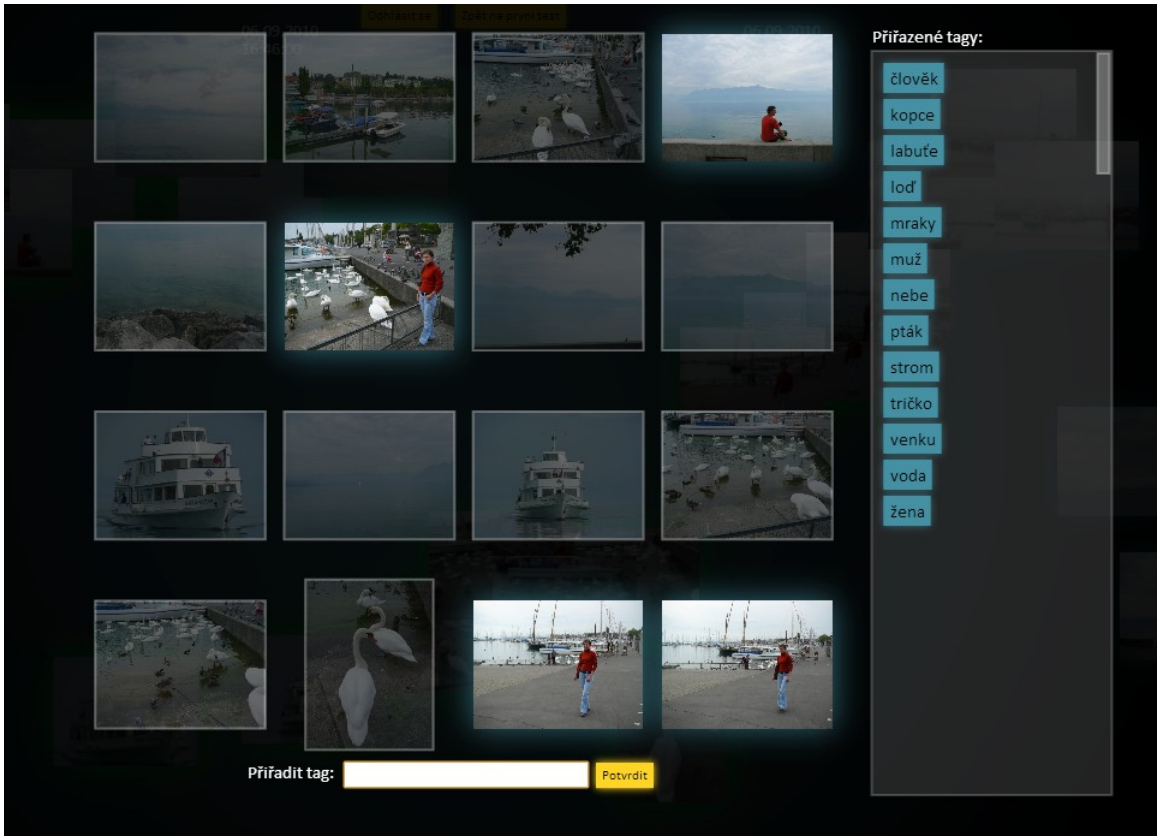
Obrázek 3.14 – snímek GUI v Timeline režimu



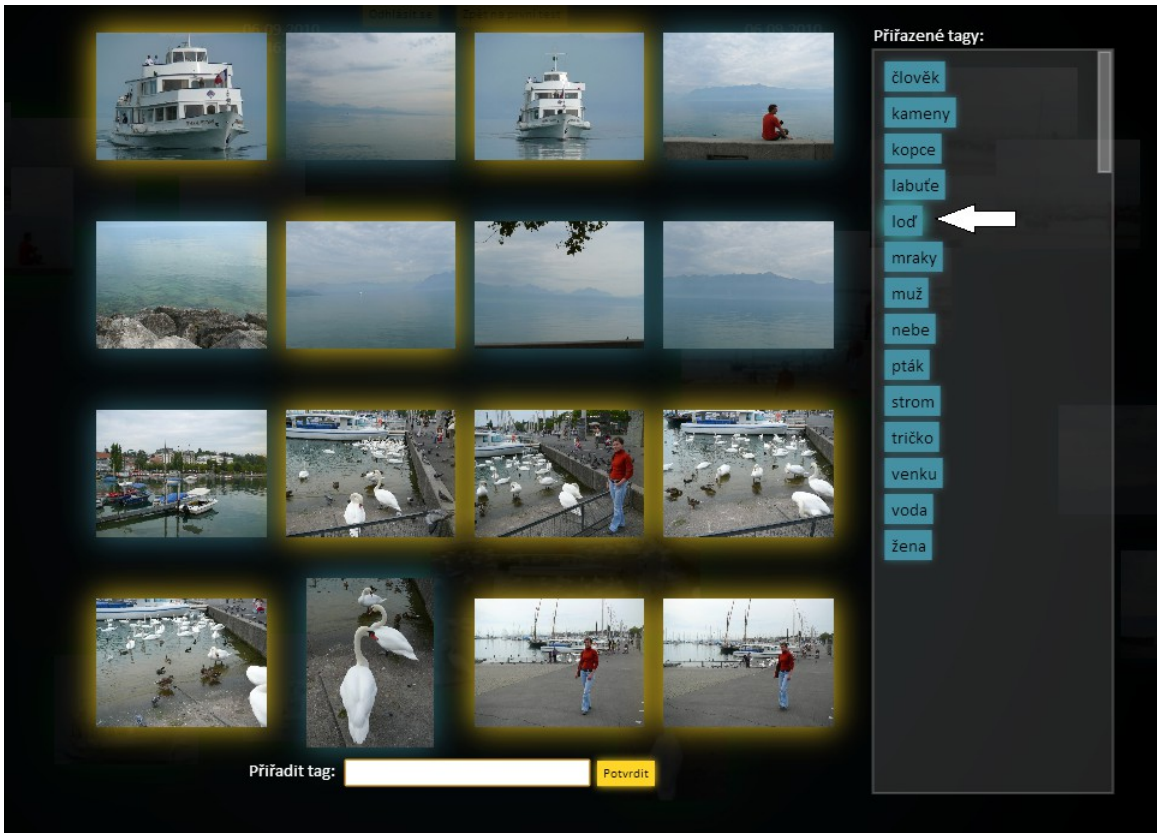
Obrázek 3.15 – snímek GUI v Timeline režimu, stejné fotografie jako na obrázku 3.14, ale po použití kolečka myši k roztažení časové osy, pro snížení překryvu fotografií, zvýšení přehlednosti a zjednodušení výběru skupin fotografií. Je dobře vidět, že projekce kvalitně oddělila fotografie s výhledem na vodní plochu (horní část vlevo), fotografie lodi (vlevo dole), labutí v přístavu (střed dole) a fotografie samotného přístavu (vpravo střed až nahoře).



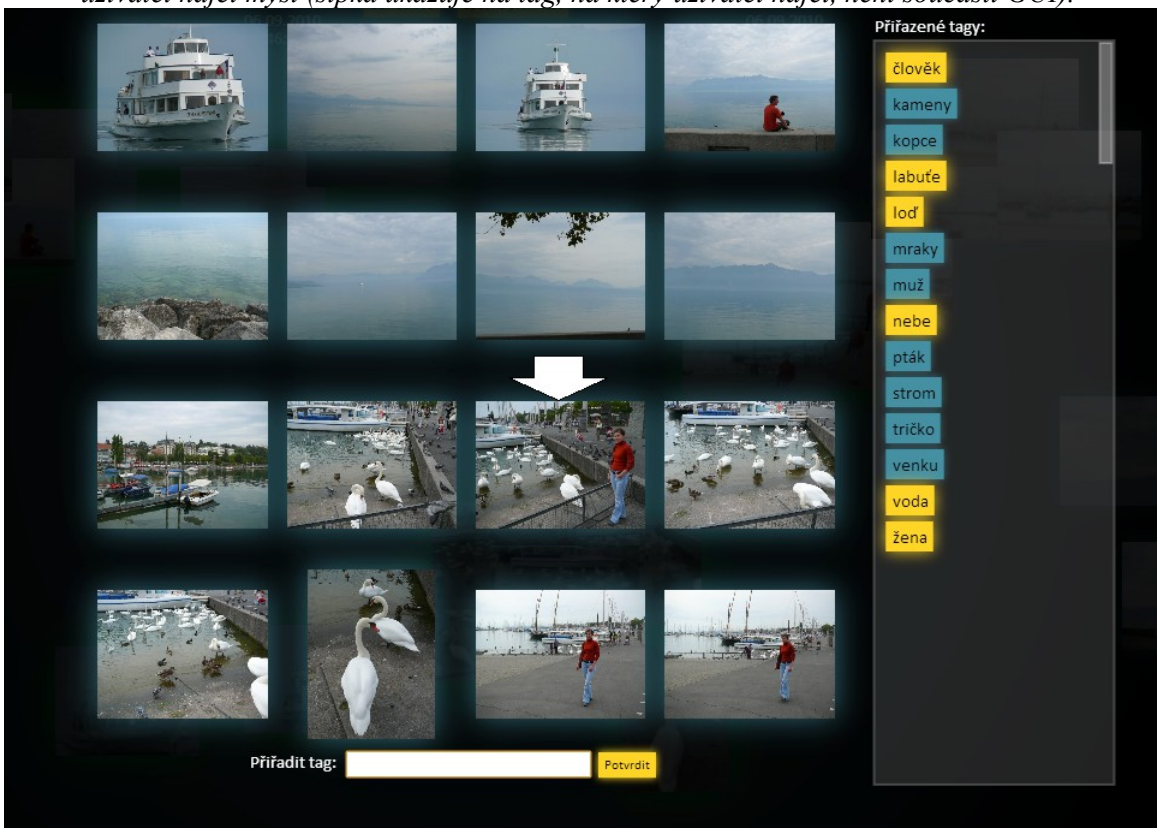
Obrázek 3.16 – snímek GUI v Tagovacím režimu. Vybrané fotografie jsou podle svého počtu uspořádány do mřížky, která se nejvíce blíží čtvercové, a je změněno jejich měřítko, aby se vešly na obrazovku všechny a zároveň byly co největší. Dole je vstupní řádek na přiřazování tagů, vpravo seznam tagů, přiřazených alespoň jedné z aktivních (tady všech vybraných) fotografií.



Obrázek 3.17 – GUI v Tagovacím režimu s aktivními jen 4 fotografiemi. Vpravo jsou vypsané jen tagy, které má přiřazené alespoň jedna z aktivních (světlejších a kontrastnějších) fotografií.



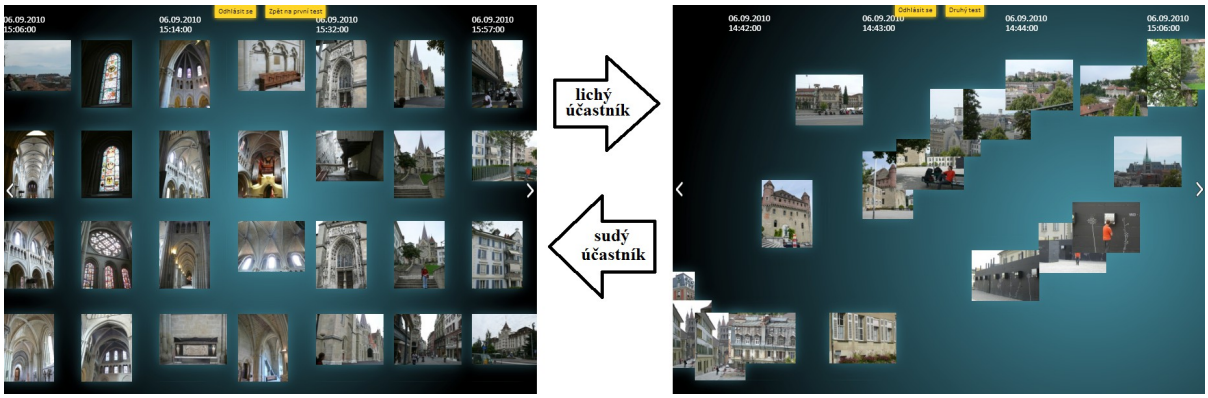
Obrázek 3.18 – GUI v Tagovacím režimu. Zvýraznění fotografií, které mají přiřazeny tag, na který uživatel najel myší (šipka ukazuje na tag, na který uživatel najel, není součástí GUI).



Obrázek 3.19 – GUI v Tagovacím režimu. Zvýrazněny tagy, které má přiřazena fotografie, na kterou uživatel najel myší (znázorněno šipkou, šipka není součástí GUI).

4 Experiment

Pro určení, zda se použití shlukování pomocí *Timeline projekce* zvýší rychlost tagování, jsem navrhl experiment, v rámci kterého 10 jedinců (5 mužů a 5 žen) pod dohledem taguje ve dvou variantách uživatelského rozhraní. Jedna varianta používá *Timeline projekci*, druhá jen poskládá fotografie do pravidelné mřížky, jak je běžné v ostatních současných tagovacích rozhraních (viz obrázek *4.1*). Vše ostatní je však mezi variantami stejné, aby experiment skutečně odhaloval přínos *Timeline projekce* jako takové. Každý z 10 účastníků pak 15 minut taguje v jedné variantě a 15 minut v druhé variantě. Každý lichý účastník začíná prací ve variantě s mřížkou a končí prací ve shlukované variantě, každý sudý začíná shlukovanou variantou a končí mřížkou, aby se statisticky co nejvíce vyloučil efekt učení, při kterém se práce jedince zrychluje, a efekt únavy, při kterém se práce jedince k závěru experimentu může zpomalovat.



Obrázek 4.1 – Dvě varianty uspořádání fotografií použité pro experiment. Liší účastníci začínají 15 minutami práce ve variantě uspořádané do pravidelné mřížky, pak pokračují 15 minutami práce v Timeline projekci. Sudí přesně naopak.

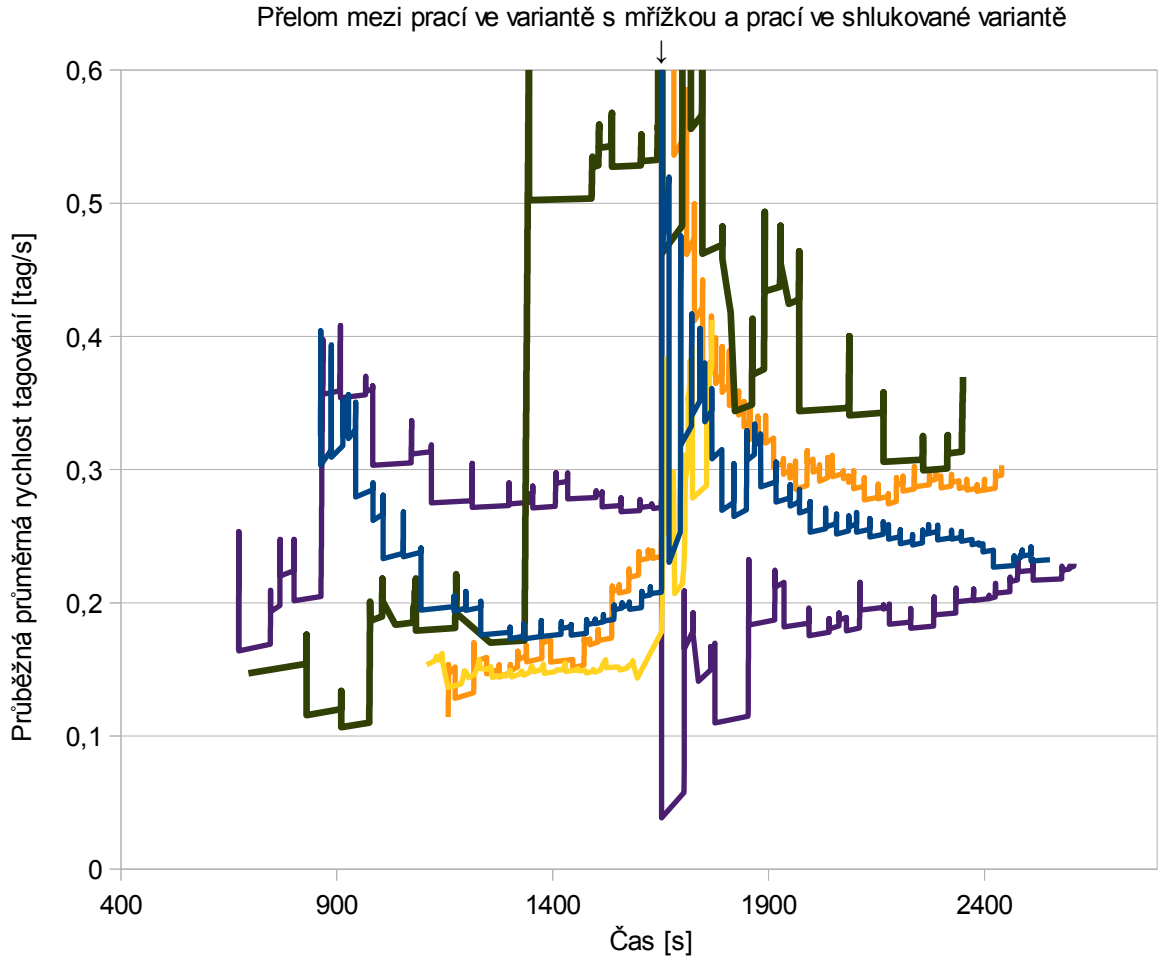
Z časových údajů v databázové tabulce *TagMembership* v databázi následně zjišťuji rychlost práce účastníků v jedné a druhé variantě, dále průměrnou velikost skupiny fotek, kterým jedinec přiřadil tag zároveň, a nakonec poměrnou změnu těchto všech parametrů mezi variantou mřížky a variantou se shluky (tedy poměrné zrychlení práce a zvětšení skupin fotografií).

Na závěr je každý zúčastněný vyzván k vyplnění krátkého dotazníku.

4.1 Naměřené hodnoty

V této části uvádím naměřené hodnoty pomocí tabulek či grafů, podle vhodnosti. Tyto hodnoty následně hodnotím.

V grafu *4.1* vidíme průběžné rychlosti tagování uživatelů skupiny *MC* (uživatelé, kteří nejprve začínali ve variantě s mřížkou, poté ve variantě s *Timeline projekcí*). Měření začalo až poté, co se účastník seznámil s uživatelským rozhraním. Skoky v grafu představují místa, kde uživatel přiřadil tag vetší skupině fotografií zároveň, což lokálně výrazně zvýšilo průměrnou rychlost. Skok v grafech na přelomu mezi variantami je dán tím, že se v tomto okamžiku i restartuje průměrná rychlost, takže každé přiřazení tagu má zpočátku větší vliv, a účastník už je seznámený s rozhraním lépe, takže pracuje již od začátku rychleji. Proto je tento skok v grafech patrnější na začátku druhé části experimentu více, než na začátku části první.



Graf 4.1 - Průběžné průměrné rychlosti práce účastníků pracujících nejdříve ve variantě mřížky, poté ve variantě s Timeline projekcí (MC).

Na grafech je patrný rozdíl mezi rychlostí práce s variantou s mřížkou a variantou se shluky, nicméně tento rozdíl se mezi účastníky liší. Účastníci, jejichž rychlost je v grafu znázorněna modrou, oranžovou a žlutou barvou, pracovali ve shlukované variantě znatelně rychleji. Účastník, jehož rychlost je znázorněna tmavě zeleně, pracoval v obou variantách v průměru přibližně stejně rychle, a účastník reprezentovaný fialovou barvou naopak pracoval ve shlukované variantě výrazně pomaleji.

Graf *4.2* obsahuje totéž měření, jako graf minulý, jen pro *CM* skupinu účastníků (účastníci, kteří pracovali nejdříve ve variantě s *Timeline projekcí*, poté ve variantě s mřížkou). Kromě implicitních artefaktů identických s grafem předchozím, je v tomto grafu patrné, že někteří účastníci pracovali ve variantě s mřížkou rychleji, než ve variantě se shluky (světle červená, světle zelená, tmavě zelená), a někteří pracovali v obou variantách přibližně stejnou rychlostí nebo o něco pomaleji (tmavě červená, světle modrá).

Graf 4.2 - Průběžné průměrné rychlosti práce účastníků pracujících nejdříve ve variantě Timeline projekce, poté ve variantě s mřížkou (CM).

Tabulka *4.1* shrnuje informace obsažené v předchozích dvou grafech. Co k těmto informacím přidává, jsou hlavně průměrné hodnoty poměrného zrychlení varianty se shluky oproti variantě s mřížkou, a to pro všechny účastníky, ať už ze skupiny *MC* či *CM*.

Dále přidává tzv. hodnoty *skupinovosti*, tedy průměrné množství fotografií, kterým účastník experimentu přiřadil jeden tag zároveň jako skupině, a k těmto hodnotám také hodnotu poměrného zvýšení ve variantě se shluky oproti variantě s mřížkou, opět pro všechny účastníky bez ohledu na skupinovou příslušnost.

Poměrná zrychlení a zvýšení skupinovosti mezi variantami nejsou příliš velká ani jedním ani druhým směrem (kromě výjimky uživatele *MC-2*), a průměrné hodnoty to jen potvrzují.

	Rychlost mřížka [tag/s]	**Rychlost shluky [tag/s]**	**Poměr zrychlení mezi mříž. a shluk. testem**	**Skupinovost mřížka [fotog./skup.]**	**Skupinovost shluky [fotog./skup.]**	**Poměr zvýšení skupinovosti mezi M a S**
MC-1	0,21	0,23	1,09	6,78	5,53	0,82
MC-2	0,14	0,41	2,87	1,06	2,45	2,32
MC-3	0,56	0,37	0,67	33,31	13,63	0,41
MC-4	0,28	0,23	0,81	13,57	8,80	0,65
MC-5	0,25	0,30	1,19	9,00	7,27	0,81
CM-1	0,26	0,18	0,68	2,42	1,70	0,70
CM-2	0,30	0,22	0,76	7,15	6,71	0,94
CM-3	0,39	0,40	1,01	1,51	1,87	1,24
CM-4	0,10	0,19	1,77	3,81	5,63	1,48
CM-5	0,42	0,21	0,50	7,02	4,02	0,57
AVG-MC	0,29	0,31	1,33	12,74	7,54	1,00
AVG-CM	0,30	0,24	0,94	4,38	3,98	0,99
AVG	**0,29**	**0,27**	**1,14**	**8,56**	**5,76**	**0,99**

Tabulka 4.1 - průměrné hodnoty naměřených rychlostí. Řádky označené kódem MC-<číslo> představují hodnoty získané pro účastníky, kteří začínali prací ve variantě s mřížkou, a ve variantě se shluky pracovali až poté (M jako Matrix, C jako Cluster). Kódem CM-<číslo> jsou označeny řádky získané pro účastníky postupující opačným směrem. Kód AVG značí průměrnou hodnotu pro daný sloupec, buď vždy pro jednu skupinu účastníků (modré řádky) nebo pro všechny účastníky celkem.

4.2 Výsledky dotazníku

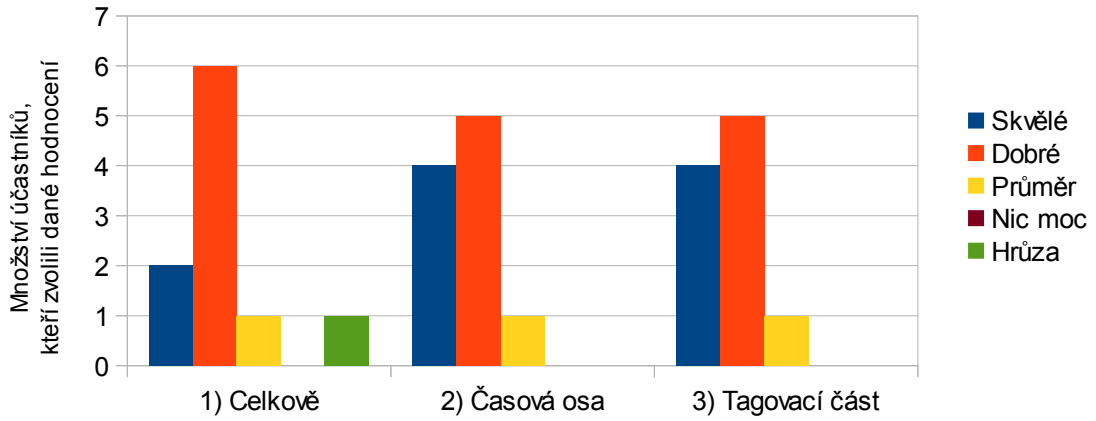
Po dokončení práce každý účastník experimentu anonymně vyplnil krátký dotazník, jehož výsledky jsou uvedeny níže formou.

Hodnotící otázky:

1) *Ohodnoťte, jak se Vám s aplikací celkově pracovalo*

2) *Ohodnoťte část rozhraní s časovou osou*

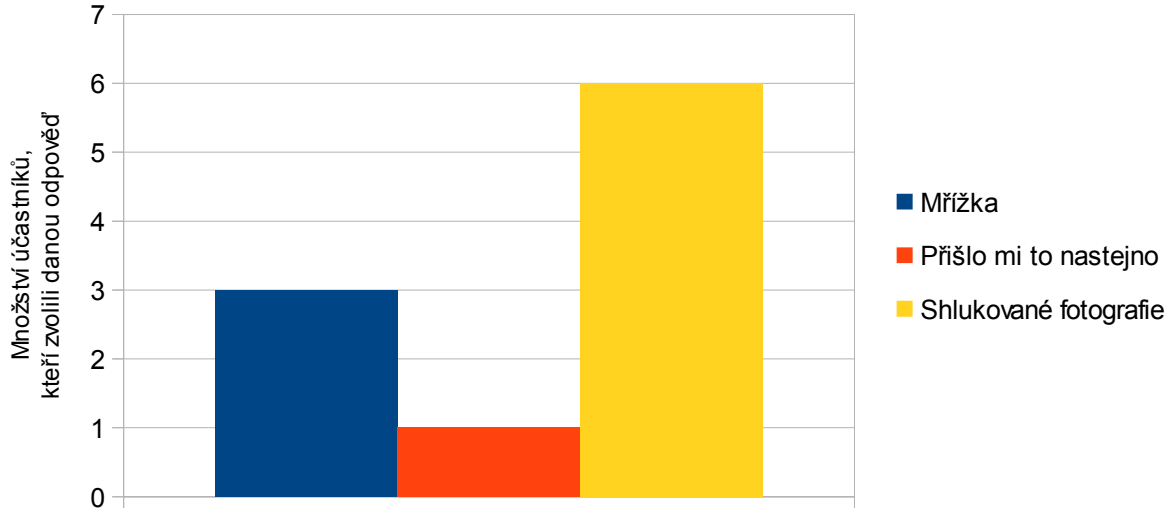
3) *Ohodnoťte tagovací část rozhraní*



Graf 4.3 – Výsledky hodnotících otázek v dotazníku.

Srovnávací otázka:

4) *S kterou variantou se Vám pracovalo lépe? Shlukované fotografie podle vizuálních aspektů, nebo obyčejná mřížka?*



Graf 4.4 – Výsledky porovnávací otázky v dotazníku.

Textová otázka:
5) Popište svými slovy, jak na Vás rozhraní působilo, co se Vám líbilo, co ne, co byste vylepšili.

Cituji některé přínosné části některých odpovědí:

1) „(...) Nápad se skupinkami fotografií, co si jsou podobné, je dobrý a zdá se mi i rychlejší, ale vzhledově na prohlížení se mi líbila první verze víc (mřížka). (…) Vícekrát se mi ale stalo, že jsem na ten použitý tag omylem klikla a tag se vymazal. To by mohlo být více komplikované, aby se to nestávalo příliš často. (...)“

2) „(...) Lépe se mi pracovalo se shluky, mohla jsem si vybrat téměř kterýkoli a věděla jsem, že tagování bude snadné. (...)“

3) „Shlukované fotografie na časové ose jsou hezčí, příjemnější na pohled. Pro výběr myší mi přišlo pohodlnější rozvržení do mřížky, ikdyž vlastně nevím proč, možná je to jen konzervatismus. Asi si chci sám třídit fotografie dle svých kritérií – pokaždé dle odlišných. Shluky to dělají místo mě a ne pokaždé to odpovídá mému třídění. (…) Kdybych si měl nakonec vybrat, bral bych shluky, jsou hezčí, přirozenější. (...)“

4) „Shlukování pro tagování je zajímavý nápad. Opravdu to hodně urychluje práci, pokud má člověk spoustu podobných fotografií a nebo dost fotografií z jednoho místa (což je v dnešní době pravda, když fotím, tak všechno 10x). Přišlo mi to trochu pomalejší (...)“

5) „(...) V tagovacím režimu by se hodily možnosti další manipulace s existujícími tagy, než jen jejich odmazání z aktivních fotografií. Hrozivě pomalé a náročné na paměť. (…) Je prostor pro zlepšení – například řazení do složek, vyhledání fotografií podle tagů (mimo tagovací režim), zobrazení tagů aktivní složky. (…) Časová osa trochu nevýrazná, mohla by se nějakým způsobem promítnout do pozadí. (...)“

Z textových odpovědí (např. citace *1, 2, 3, 4*) vyplynulo, že i když se většině účastníků více líbila shlukovaná verze zobrazení fotografií, tak ne vždy to je to, co by od podobné aplikace požadovali, nehledě na to, že právě na klasické zobrazení do mřížky jsou všichni zvyklí z jiných aplikací, proto měli menší problém se s ní naučit pracovat, oproti shlukovaným fotografiím. Na druhou stranu většina nápad se shlukováním fotografií hodnotila kladně.

Dále z této otázky vyplynuly další náměty pro úpravy uživatelského rozhraní (např. citace *1, 5*), z těch nejvěcnějších například aby vstupní řádek nabízel seznam uživatelem již použitých tagů, které začínají řetězcem zapsaným ve vstupním řádku, dále možnost kliknutím na přiřazený tag jej přiřadit úplně všem aktivním fotografiím, možnost opravy textu tagu při překlepu, aniž by bylo potřeba jej mazat a znovu zadávat, upravení odběru tagu, aby se zamezilo nechtěnému odebrání, a mnohé další.

V neposlední řadě se v odpovědích na tuto otázku projevilo, že si někteří účastníci povšimli nepříliš vysoké rychlosti uživatelského rozhraní (např. citace *4,5*) – teď není řeč o rychlosti načítání rozhraní a postupného načítání fotografií při procházení kolekcí, ale o samotné rychlosti reakcí rozhraní v rámci již načtených fotografií. Z tohoto lze odvodit, že použitá zobrazovací knihovna *KineticJS* není příliš vhodná pro zobrazování většího množství prvků, zejména fotografií, kdy už začíná být odezva příliš pomalá. Nutno připomenout, že fotografie mimo část, která je aktuálně na obrazovce a nejbližší okolí, jsou skrývány, takže při zobrazení je většinou počítáno jen asi s 60 malými náhledy fotografií (největší hrana každého fyzického náhledu má 120px, průměrně 7KB dat na 1 náhled fotografie), což není zas až tak velké množství dat, při kterém by měla práce na průměrném stroji být pomalá.

4.3 Vyhodnocení

Z naměřených hodnot nevyplynulo nic jednoznačného – jsou zde dvě možnosti. Jedna zní, že v tak krátkém časovém období, jakým je 15 minut, si pravděpodobně uživatel není schopen dostatečně osvojit práci s uživatelským rozhraním, aby byly opravdu jasněji patrné rozdíly v rychlosti práce mezi dvěma testovanými variantami, případně díky časovému omezení není uživatel schopen zpracovat dostatečné množství fotografií, aby se neprojevila nekonzistence mezi charaktery fotografií použitých v první a druhé části testu. Druhou možností je fakt, že obě varianty jsou podobně vhodné, co se týče rychlosti tagování, tedy že se u obou najdou jedinci, kteří ve shlukované variantě pracují rychleji, než v mřížce, a také jedinci opační.

Pro určení, která z těchto možností je důvodem pro výsledek mého experimentu, by bylo potřeba rozsáhlejšího experimentu, a to jak časově (podstatným způsobem), tak i počtem účastníků.

Nicméně pokud by platila druhá možnost, že *Timeline projekce* i obyčejná mřížka jsou stejně vhodné, nebo se jejich vhodnost liší osobu od osoby či kolekci fotografií od kolekce, pak se nabízí možnost hybridního rozhraní, ve kterém by se ve fázi projekce připravila jak *Timeline projekce*, tak obyčejná mřížka, a mezi těmito variantami by si uživatel mohl libovolně přepínat nad jednou kolekcí fotografií, podle své obliby a podle vhodnosti vůči dané kolekci. Toto rozšíření lze vytvořit i plynulé, tedy kdy uživatel může do jisté míry plynule měnit váhu časového seřazení fotografií a shlukování podle vizuálních aspektů. Pro tyto účely by ale bylo potřeba alespoň extrémy (mřížka a *Timeline projekce*) předpočítat předem, a pozice fotografií by bylo možno mezi těmito extrémy interpolovat. Pokud by bylo potřeba nastavovat váhu všech parametrů *Timeline projekce* samostatně (barevné histogramy, expozice, ISO, clona, cyklický čas, absolutní čas), pak by bylo potřeba buď předpočítat alespoň všechny kombinace extrémů těchto vah a z nich následně interpolovat, a nebo použít jinou úpravu projekce t-SNE, která podporuje lokalizaci výpočtů, aby výpočet projekce netrval příliš dlouho a mohl se provádět online podle požadavků uživatele. V současné době projekce 700 fotografií trvá přibližně 3 minuty, bez načítání vstupu 2 minuty (na Intel Core 2 Duo 1,66 Ghz, 2 GB RAM), což je pro plynulé online promítání příliš dlouhá doba.

Dále z dotazníků vyplynuly potřeby některých úprav stávající části rozhraní (viz kapitola *4.2*), které je navíc pro použitelnost v reálu potřeba rozšířit o zbytek funkčnosti – uživatelsky příjemnější nahrávání fotografií na server, vytváření a správa kolekcí, možnost zadávat nad kolekcí/kolekcemi dotazy vyhledání podle přiřazených tagů.

V neposlední řadě z dotazníků (i z mého osobního pozorování) vyplynulo, že by pro reálné nasazení bylo potřeba změnit způsob zobrazování, protože knihovna *KineticJS* zatím v současné době pravděpodobně není dostatečně optimalizována pro práci s větším množstvím obrazových dat. Vhodnější bude buď implementace nativní aplikace, nebo v případě webové aplikace použít jinou zobrazovací knihovnu, či rovnou přistupovat přímo k HTML5 prvku *canvas*.

5 Závěr

Prozkoumal jsem známé přístupy k tagování fotografií. Zaměřil jsem se především na zobrazování fotografií jejich uspořádáním podle obrazových a jiných vlastností. Takto uspořádané fotografie usnadňují výběr skupin fotografií, kterým budou přiřazeny stejné tagy.

Podařilo se mi implementovat prototyp webového rozhraní, které dokáže zobrazit kolekci fotografií promítnutých pomocí mnou upravené projekce t-SNE tak, aby byly fotografie zároveň seřazeny podle času pořízení a zároveň seskupeny podle ostatních vlastností *(Timeline projekce),* např. podle barevných histogramů. Převzatou projekční metodu t-SNE jsem pro tyto účely upravil několika různými způsoby, otestoval je a vybral jsem z nich ten nejvhodnější z hlediska výpočetní náročnosti, přínosu pro seskupení podobných fotografií kolekce a pro efektivitu tagování fotografií v kolekci.

Implementovaný prototyp uživatelského rozhraní pro efektivní tagování jsem podrobil experimentu pro srovnání rychlosti práce s fotografiemi uspořádanými pomocí *Timeline projekce* proti fotografiím uspořádaným do jednoduché mřížky. Tento experiment však nepřinesl průkazné výsledky jedním ani druhým směrem, pravděpodobně by byl potřeba rozsáhlejší experiment, zejména časově.

Pokud by se prokázalo, že je efektivita jedné či druhé varianty rozdílná pro jednotlivé uživatele a kolekce fotografií, pak jsem v kapitole *4.3* spolu s vyhodnocením a možnostmi dalšího pokračování výzkumu uvedl možnost hybridního rozhraní, ve kterém by se plynule přepínalo mezi mřížkou a *Timeline projekcí* pomocí nastavování vah parametrů projekce.

Literatura

[1] P. Berkhin: Survey of clustering data mining techniques, Technical report, Accrue Software, San Jose, CA, 2002

[2] P. Browne, A. Smeaton: Video information retrieval using objects and ostensive relevance feedback, In: ACMsymp applied computing. ACM, New York, 2004, pp. 1084-1090

[3] K. Cox: Information retrieval by browsing, In: Proc int'l conf new information technology, Hong Kong, 1992

[4] K. Cox: Searching through browsing, PhD thesis, University of Canberra, 1995

[5] R. Duda, P. Hart, D. Stork: Pattern recognition, New York, Wiley, 2001

[6] D. Heesch: The NNk technique for image searching and browsing, PhD thesis, Imperial College London, 2005

[7] D. Heesch: A survey of browsing models for content based image retrieval, In: Multimedia Tools and Applications, vol. 40, no. 2, 2008, pp. 261–284

[8] G.E. Hinton, S.T. Roweis: Stochastic Neighbor Embedding, In: Advances in Neural Information Processing Systems, Cambridge, MA, USA, The MIT Press, vol. 15, 2002, pp. 833–840

[9] H. Hotelling: Analysis of a complex of statistical variables into principal components, In: Journal of Educational Psychology, 1933, pp. 417–441

[10] M. Hradiš, I. Řezníček, K. Behúň: Semantic Class Detectors in Video Genre Recognition, In: Proceedings of VISAPP 2012, Rome, IT, SciTePress, 2012, pp. 640-646

[11] R. Jacobs: Increased rates of convergence through learning rate adaptation, In: Neural networks, 1988

[12] I. Keller, T. Meiers, T. Ellerbrock, T. Sikora: Image browsing with PCA-assisted user-interaction, In: IEEE workshop content-based access of image and video libraries, IEEE, Piscataway, 2001, pp. 102–108

[13] L. Van Der Maaten, G. Hinton: Visualizing data using t-SNE, In: Journal of Machine Learning, vol. 9, 2008, pp. 2579–2605

[14] S.T. Roweis, L.K. Saul: Nonlinear dimensionality reduction by Locally Linear Embedding, In: Science, vol. 290, no. 5500, 2000, ppp. 2323–2326

[15] J.W. Sammon: A nonlinear mapping for data structure analysis, In: IEEE Transactions on Computers, vol. 18, no. 5, 1969, pp. 401–409

[16] J.B. Tenenbaum, V. de Silva, J.C. Langford: A global geometric framework for nonlinear dimensionality reduction, In: Science, vol. 290, no. 5500, 2000, pp. 2319–2323

[17] W.S. Torgerson: Multidimensional scaling I: Theory and method, In: Psychometrika, vol. 17, 1952, pp. 401–419.

[18] J. Vendrig, M. Worring, A. Smeulders: Filter image browsing: exploiting interaction in image retrieval, In: Visual information and information systems, Amsterdam, 1999, pp. 147–154

Seznam příloh

Příloha 1. CD se zdrojovými texty implementovaných aplikací